Improving Traffic Flow by Local Methods

Von der
Carl-Friedrich-Gauß-Fakultät
der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
genehmigte Dissertation
von
Dipl.-Phys. Björn Alexander Hendriks
geboren am 2. Oktober 1968
in Oberhausen

Eingereicht am: 8. März 2013
Disputation am: 17. April 2013
1. Referent: Prof. Dr. Sándor P. Fekete
2. Referent: Prof. Dr. Stefan Fischer

2013

Abstract

For many decades congestion on highways and arterial roads has been a major problem of vehicular traffic worldwide. It is not possible to build and extend roads to keep up with the increase of traffic. Traffic congestion has a major impact on the environment, the economy and last but not least on the individual driving comfort. Congestion is caused by large-enough perturbations in dense traffic.

In recent years car manufacturers and independent researchers started to develop in-vehicle devices to automatically establish wireless communication among equipped vehicles in the vicinity, known as *Car-to-Car* (C2C) communication. Often announced use cases of such an ad-hoc network are emergency warnings, traffic information (to avoid congested road sections), and internet access.

We developed a driving strategy using C2C communication to prevent congestion, which we call *Jam-ADS* (ADS = Advanced Distributed Strategy). C2C communication provides information about current positions and velocities of the vehicles ahead to a Jam-ADS processor in each vehicle. Jam-ADS aggregates this information and computes a velocity recommendation, which is presented to the driver. If sufficiently many drivers follow the recommendations congestion does not occur.

In this work, we tested and evaluated several variants of Jam-ADS by simulating different scenarios. To do this, we developed our own traffic simulator *CircSim* to be able to influence the simulation and freely evaluate the results. CircSim simulates a closed road system consisting of a circular road of arbitrary length with arbitrary many lanes. To validate the results, we performed the same simulations with the established open-source traffic simulator SUMO and the communication network simulator Shawn.

To make the simulations more realistic, we also performed our simulations in an open system consisting of a section of a highway with an on-ramp, because on-ramps are a major source of congestion-causing perturbations. Such a scenario is often used for the simulation of traffic congestion.

In both scenarios we performed simulations, which showed that Jam-ADS improves the flow of traffic.

To support the effectiveness of Jam-ADS beyond numerical simulations, we analyzed it mathematically as well by applying methods of control engineering. They confirmed that Jam-ADS damps all perturbations.

Zusammenfassung

Schon seit einigen Jahrzehnten sind Staus auf Autobahnen und Hauptverkehrsstraßen ein großes Problem im Straßenverkehr, da es nicht möglich ist, so schnell und umfangreich neue Straßen zu bauen oder bestehende zu erweitern, wie die Anzahl der Fahrzeuge steigt. Staus haben bedeutenden Einfluss auf die Umwelt, die Wirtschaft und nicht zuletzt auf den individuellen Fahrkomfort. Im dichten Verkehr können schon geringe Störungen einen Stau verursachen.

In den letzten Jahren haben Automobilhersteller und unabhängige Forscher begonnen, Geräte zu entwickeln, die automatisch ein drahtloses Kommunikationsnetz zwischen allen mit dem Gerät ausgerüsteten Fahrzeugen aufspannen, sofern sie nahe genug sind, welches *Car-to-Car*-Kommunikation (C2C) genannt wird. Häufig werden als Anwendungsfälle Notfallwarnungen, Verkehrsinformationen (um Staus zu umfahren) und Internetzugang genannt.

Wir haben eine Strategie namens *Jam-ADS* (ADS = Advanced Distributed Strategy) entwickelt, die diese C2C-Kommunikation nutzt, um Staus und stockenden Verkehr zu verhindern. C2C-Kommunikation liefert laufend Informationen über Positionen und Geschwindigkeiten von vorausfahrenden Fahrzeugen an den Jam-ADS-Prozessor in jedem Fahrzeug. Jam-ADS fasst diese Informationen zusammen und berechnet daraus eine Geschwindigkeitsempfehlung, die dem Fahrer angezeigt wird. Wenn hinreichend viele Fahrer und Fahrerinnen der Empfehlung folgen, tritt kein Stau auf.

In dieser Arbeit haben wir mehrere Varianten von Jam-ADS durch Simulationen von verschiedener Szenarien getestet und ausgewertet. Dafür haben wir einen eigenen Verkehrssimulator namens *CircSim* entwickelt, um frei die Simulation zu beeinflussen und die Ergebnisse auszuwerten. CircSim simuliert ein geschlossenes Straßennetz bestehend aus einer Ringstraße beliebiger Länge mit beliebiger Anzahl Spuren. Um die Ergebnisse zu validieren, haben wir die gleichen Simulationen auch mit dem etablierten quelloffenen Verkehrssimulator SUMO und dem ebenfalls quelloffenen Kommunikationsnetzwerksimulator Shawn durchgeführt.

Um die Simulationen realistischer zu gestalten, haben wir weitere Simulationen mit einem offenen System bestehend aus einem Autobahnabschnitt mit einer Einfahrt durchgeführt, da an Einfahrten häufig stauauslösende Störungen entstehen. Dieses Szenario wird in der Verkehrsforschung häufig benutzt, um Staus zu simulieren.

Mit beiden Szenarien führten wir zahlreiche Simulationsläufe durch, welche zeigten, dass Jam-ADS den Verkehrsfluss verbessert und so Staus vermeidet.

Schließlich haben wir Jam-ADS auch mathematisch analysiert, um die Effektivität von Jam-ADS auch jenseits von numerischen Simulationen nachzuweisen. Mithilfe von etablierten Analysemethoden aus der Regelungstechnik haben wir nachgewiesen, dass Jam-ADS geeignet ist, Störungen zu dämpfen.

Acknowledgments

First of all, I would like to thank **Prof. Dr. Sándor P. Fekete** for giving me the chance to acquire a doctorate, although I was not the typical applicant with a recent, closely related, academic degree. He provided me with an interesting project and lots of useful advice to successfully work on it and gave me the freedom to go my own way to achieve results.

Together with Sándor Fekete I would like to thank **Prof. Dr. Stefan Fischer** for successfully applying to the DFG priority program 1183 Organic Computing to support project AutoNomos including my work. I also thank them and the other participants of project AutoNomos, **Prof. Dr. Horst Hellbrück**, **Dr. Axel Wegener**, and **Sebastian Ebers** for lots of fruitful discussions and enjoying together the periodical meetings of the priority program.

Additionally, **Dr. Christiane Schmidt** and **Christopher Tessars** contributed a lot to project AutoNomos before I stepped in. Particularly, I would like to thank them for providing the basic idea of the strategy mainly investigated in this work. Christiane Schmidt also helped in proofreading of this thesis.

Special thanks go to my office mate **Dr. Chris M. Gray** with whom I had many interesting conversations, both scientific and private, and who shared my preference of a late lunch. As native English speaker he helped me to improve my English skills and also with proofreading of this thesis. I also thank everybody else who took the time to suggest corrections and improvements for this thesis, mainly **Dr. Iris Reinbacher**, but also **Henning Hasemann** and **Marina Tvorogova**.

I thank all current and former members of the algorithms group not mentioned yet, **Dr. Tobias Baumgartner**, **Ingo Brinkmeier**, **Stephan Friedrichs**, **Dr. Tom Kamphans**, **Prof. Dr. Alexander Kröller**, **Andreas Landau**, **Martin Lorek**, **Dr. Mahdi Moeini**, **Max Pagel**, **Dr. Nils Schweer** for all the time we shared and enjoyed.

I also thank **Sabine Anthony**, **Frank Steinberg**, and **Ulrich Timm** for their personal, administrative, and technical support. Sabine Anthony was always ready with an open ear for my needs. Frank Steinberg and Ulrich Timm did a great job to keep the computation servers running to produce tons of simulation data in endless hours of full processor load. Frank Steinberg was always prepared to fulfill my numerous wishes to change the configuration of the servers.

Finally and most of all, I thank my partner **Susanne** for supporting me and tolerating all the extra time I put into working on this thesis.

Contents

1. Introduction

It is Friday afternoon, you drive home after a long week eagerly looking forward to a relaxing weekend. Unfortunately, many other drivers have the same idea. Soon after you started you are stuck in a massive traffic jam and your destination seems to be farther than ever. Now imagine, instead of a traffic jam around you, you have a special speedometer with a second small hand pointing sometimes to a speed slightly less than your current desired speed. You follow this recommendation and arrive at home without having to work through any congestion.

This work develops a strategy which helps to make that vision come true. The strategy does this with very few resources. It only applies wireless communication to the vehicles in vicinity that are similarly equipped. Beside that, the strategy only needs vehicle-internal data like current velocity and GPS position.

In the literature, wireless communication among vehicles is called *car-to-car communication* (C2C). Communication to fixed devices installed at the road—known as road-side-units (RSU)—is called *car-to-infrastructure communication* (C2I). Wireless communication in general between cars and any entities including other vehicles is denoted as *C2X communication*. Technology based on C2X communication is denoted as *intelligent transport system* (ITS). The physical and link layers of the OSI model are already standardized for C2X in the IEEE standard P802.11-2012, which incorporates the previous standard P802.11p.

Today, traffic is usually measured by induction loops built into the road. Eckehard Schnieder mentioned in a public talk that within their typical distance of 1 km to 1.5 km a vehicle may accelerate and decelerate several times, thus according to the Nyquist-Shannon sampling theorem they are too far apart to properly sample traffic oscillations. C2C communication can provide a much denser sampling than induction loops.

We limit our strategy to C2C communication without any C2I communication. Every C2X communication application needs vehicles to be equipped with wireless devices, but the additional equipment of roads with RSUs is more expensive and it will take a long time until all roads are equipped with RSUs if ever. On the other hand, pure C2C applications can operate everywhere once enough vehicles are equipped with C2C devices.

The problem of traffic congestion is growing worldwide. In 2010, a traffic jam more than 100 km long occurred in China, which lasted more than ten days. Of course, this is an extreme example, but numerous studies on congestion costs indicate the problem. Several estimates of congestion costs in Great Britain were

collected by Goodwin [42] ranging up to £20 billion per year at the end of the 1990s, though he noted that there are many uncertainties in estimating congestion costs. An Australian study [13] assesses the congestion costs in 2005 at $9.4 billion and estimates that they will reach up to $20.4 billion by 2020. The Canadian Greater Toronto Transportation Authority [45] estimates the congestion costs of Greater Toronto and Hamilton Area, Ontario, Canada in 2006 to be $3.3 billion for commuters and $2.7 billion for reduction of the GDP (Gross Domestic Product). They expect in 2031 costs of $7.8 billion for commuters and $7.2 billion for reduction of the GDP if no measures are taken. The considered causes of costs are reduced economic output including job loss, travel delays including extra time because of unreliability of trip times, increased vehicle operating costs, increased pollution, and additional accidents. Schrank, Eisele, and Lomax [129] calculate the additional time spent in congestion in 498 American urban areas in 2011 to be 5.5 billion hours, consuming 2.9 billion gallons of additional fuel, causing additional costs of $121 billion for fuel and delay time. Per commuter, these are costs of $818 and 38 hours in congestion compared to $342 (inflation-adjusted) and 16 hours in 1982. The study also found that congestion is a problem in non-peak hours, because many manufacturers shift their transportation demand to these times to avoid peak congestion, making non-peak times also less reliable. The Congestion Index of the navigation device manufacturer TomTom annually provides congestion data of big cities worldwide, see for example [135] for European cities in 2012. All these and many more studies illustrate that traffic congestion is already a major problem and it will become worse if nothing is done.

Currently many different strategies to avoid or mitigate congestion on highways are developed and tested around the world. Chapter 4 gives an overview. Papageorgiou et al. [112] argue that uncontrolled freeway traffic resembles urban traffic before traffic lights were introduced to control urban traffic flows. In those years, it took many different attempts to develop a design for useful traffic lights until they became standardized in the 1920s [105].

A reason for the occurrence of congestion on highways is the emergence of a Nash equilibrium [171]. In a Nash equilibrium no individual can reduce his/her personal costs on his/her own, but there exists a global optimum at which the total costs are less than in the Nash equilibrium. The ratio of the total costs in the global optimum and the Nash equilibrium is known as price of anarchy. Costs in terms of traffic are usually travel times.

Thus, traffic congestion is a phenomenon that arises naturally from the interest of each driver to choose his/her individually best route and the limited flow capacity of available roads. The first idea to solve this is to increase the capacity of crowded roads. However, that approach is expensive and in dense urban areas often impossible because of limited space. It is also not feasible to force some drivers to take a detour such that other drivers can enjoy free traffic on the direct path.

On a given road, congestion emerges because of perturbations in too dense traffic. In dense traffic a perturbation increases the velocity variance, which causes braking maneuvers that further increase the velocity variance [141]. This fulfills the definition of emergence of Wolf and Holvoet [168] such that emergence arises on the macro-level from dynamic interaction of the parts on the micro-level. Therefore, to prevent emergence we need to change interaction dynamics on the micro-level. In case of traffic, computer science can change the interaction dynamics of vehicles by providing additional information derived from C2C communication. C2C communication increases the knowledge that an individual driver has on the traffic situation ahead, but it is not sufficient to continuously pass detailed data received from hundreds of vehicles ahead directly to the driver as he/she would not be able to cope with such an amount of data. Instead, we need to aggregate these data to a simple and easy to follow driving recommendation. This thesis proposes, tests by simulation, and theoretically confirms a strategy to derive a velocity recommendation from the position and velocity data of C2C-equipped vehicles ahead, with the aim of avoiding congestion in critically dense traffic.

The thesis is divided into the following chapters. Chapters 2 to 4 give an overview over the state of research: Chapter 2 presents scientific knowledge about the forming of traffic congestion, Chapter 3 illustrates the state of the art in modeling traffic on the microscopic level, and Chapter 4 explains current measures to control traffic and the current research on C2C-based strategies. Chapter 5 presents the basic concepts and ideas of the DFG project AutoNomos [106, 124], which provided the context of this work. One of these concepts is the strategy investigated in this work. Chapter 6 describes the simulation software developed and applied for testing the strategy. Chapters 7 and 8 present simulation results of a closed and an open system, respectively. In a closed system a fixed set of vehicles circulates while in an open system at any time vehicles can enter or leave the road network. Chapter 9 explains results of a mathematical analysis of our strategy. Chapter 10 illustrates some thoughts to develop a similar driving strategy to improve urban traffic flow. And finally, Chapter 11 summarizes the results and suggests further research. A few appendices explain some details more closely.

2. Traffic: Empirical Observations

To improve traffic flow we first have to understand its mechanisms. Research on traffic flow goes back to the 1930's [91] and accumulated in the last two decades with the growing problem of congestion and also with the availability of feasible computer simulations. General overviews of empirical results are given, for example, in Chowdhury, Santen, and Schadschneider [17] and Helbing [51].

One problem of empirical observation is to gather the traffic data itself. For a long time this was done by photography from the side of a road or from the air [143]. Starting in the 1960s, many highways were equipped with *induction-loop detectors*. The older single induction-loop detectors measure the times when a vehicle enters and leaves the detector. The newer double induction-loop detectors additionally measure the velocity and the length of each passing vehicle. Since the 1980s fixed video cameras are also applied to measure traffic [49]. Video cameras are easier to install and maintain but less reliable in bad weather conditions.

An important decision when evaluating traffic data is the length of the time interval to average over [83, 109]. A too short interval may hide interesting traffic features under statistical noise while a too long interval may average out interesting details. Often used time intervals range from 1 min to 5 min.

Beside measuring real traffic, some researchers conducted experiments with drivers on circular test tracks. Those allow to control many aspects of the settings including instructions to drive in a certain manner, but are rather limited compared to the numbers of cars and sizes of usual congestion on real highways. In addition, even experienced drivers cannot exactly reproduce their behavior under the same conditions.

2.1. Fundamental diagram

Personal experience tells us that congestion depends on the amount of cars on the road. Of course, traffic jam always arises if the road is completely blocked, regardless of the number of cars on the road. However, we exclude this case, as there is no way to improve the flow on a given road if it is externally blocked, except to remove the blockage, which is beyond the scope of this work.

The basic quantities to measure the amount of vehicles on the road are flow and density. *Flow* is defined as the number of vehicles per time unit passing a given location. *Density* is the number of vehicles per unit road length. When measuring,

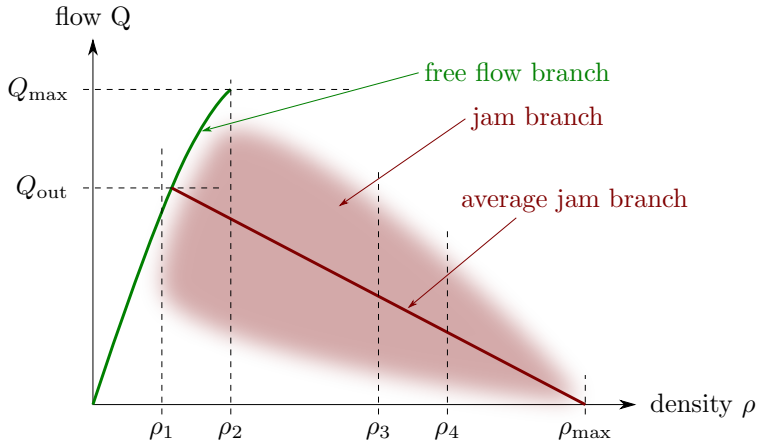


Figure 2.1.: Structure of empirical fundamental diagrams.

these quantities are averaged over a given time or road section. Although vehicles are discrete entities, some traffic theories assume them to be continuous functions of time and position as, for example, the macroscopic Lighthill-Whitham-Richards model presented in Section 3.1.

The relationship between flow and density constitutes the traffic situation. A concrete pair of flow and density is often denoted as *traffic state*. Observed traffic states are usually drawn as *fundamental diagram* of traffic research showing flow over density. Figure 2.1 is a schematic drawing of the usual empirical structure of the fundamental diagram.

All empirical fundamental diagrams consist of two distinguishable structures called *branches*. These branches form a triangular shape between the origin, a maximum flow $Q_{\max}$, and a maximum density $\rho_{\max}$ at or around zero flow. On the left is the sharp branch of free traffic (green in Figure 2.1) and on the right the cloud-like branch of traffic jam (red in Figure 2.1). The free-traffic branch forms a sharp and nearly linear structure intersecting the origin, while the jam branch consists of points scattered over a wider area. To handle the multitude of jam states, many traffic theories and models aggregate the jam branch to a line of average states.

The relation of traffic flow to density provides velocity information; in fact every slope in the fundamental diagram can be interpreted as velocity. The quotient Q/ρ equals the average velocity of the vehicles in the state with flow Q and density ρ. The maximum slope of the free-traffic branch is equal to the maximum average velocity $v_{\max}$ on the given road. For high densities the average velocity of free

traffic is a little smaller than the average velocity at lower densities, thus the free-traffic branch has a slight rightward curve at its upper part. Recent results of Coifman and Kim [20] have shown that downstream of a bottleneck the free-traffic branch may become entirely straight.

If the density of free traffic rises above the critical density ρ_1 there is a growing probability that congestion arises. At the critical density ρ_2 this probability is one. In other words, the closer the density is to ρ_2, the smaller a perturbation needs to be to cause congestion. It follows that the maximum $Q_{\max}$ of the jam branch is the maximum possible flow on the given road. Neubert et al. [109] have found that the free flow close to the maximum flow comes from fast moving platoons with risky small gaps between the vehicles, which explains that small perturbations are sufficient to cause congestion.

Congested traffic states between ρ_1 and ρ_2 have significantly less flow than the corresponding free-traffic states. Hence, the *capacity*, i.e., the maximum flow, of a congested road is less then the capacity of the same road at free traffic. Treiber and Helbing [137] mention a capacity drop between 5 % and 30 %. Thus, after the transition from free traffic to congestion, it is not possible to immediately return to the free-traffic state. On the contrary, the density has to be reduced to return to free traffic. This is called the *hysteresis effect* of the fundamental diagram. Bando et al. [3] have investigated the hysteresis on a circular single-lane road. Also, Brilon, Geistefeldt, and Regler [10] suggest a new dynamic definition of capacity.

According to Schönhof and Helbing [127], both the density region between ρ_1 and ρ_2 and the region between ρ_3 and ρ_4 are metastable, because in these regions a perturbation must have a minimum size to cause a breakdown. The region between ρ_2 and ρ_3 is linearly unstable, because here arbitrarily small perturbations can cause traffic breakdowns. Above ρ_4 we have stable congested traffic. See also Nagel, Kayatz, and Wagner [107].

The slope of the average jam branch line corresponds to the velocity of traffic jam fronts as shown in Section 3.1. It is negative, because traffic jams travel upstream (see next section). Kerner and Rehborn [79] and Kerner [68] have confirmed this empirically.

The maximum density $\rho_{\max}$ corresponds to a traffic jam of vehicles standing bumper-to-bumper (plus a minimum gap). Thus, $\rho_{\max}$ equals the inverse of the average vehicle length plus minimum gap.

Many different ways have been suggested to model the shape of the fundamental diagram. See Wang et al. [149] for a review of such models in comparison to empirical data. Wu [169] explains the shape of the fundamental diagram with mixtures of free vehicles, bunched convoys and standing vehicles. The free traffic branch consists of free vehicles and convoys, and the jam branch of convoys and standing vehicles. This approach allows to derive the concrete shape of the branches from five parameters: the desired velocity, the temporal distances of convoys in either

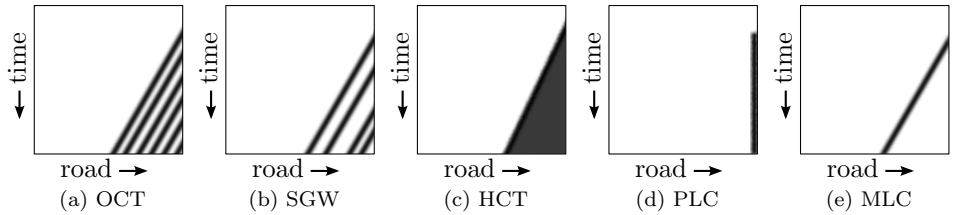


Figure 2.2.: Traffic jam types according to Schönhof and Helbing [127].

free traffic or traffic jams, the mean velocity of free traffic convoys, and the jam density. He supports his findings by empirical data.

Kim and Keller [83] assign different traffic patterns to different regions of the fundamental diagram and investigate empirically the transitions between these patterns.

2.2. Traffic jams

As described in the last section, a traffic jam emerges if the vehicle density is—at least locally—too high (higher than ρ_1 in Figure 2.1) and a large enough perturbation occurs. Road stretches of increased density with respect to the capacity of the road are bottlenecks of all kinds. Possible bottlenecks can be reduction of the number of lanes, traffic accidents, tunnels, bridges, curves, ascending slopes, or on-ramps because of the additional inflow. If the general density is high enough, off-ramps and descending slopes could be congestion causing sources of perturbation, as well.

Kerner and Rehborn [78] discovered that wide moving jams[1] can travel upstream for a long time keeping their form. Furthermore, they found that the outflow Q_{out} of wide moving jams is always approximately the same and corresponds to the intersection of the free-traffic branch and the average jam branch of the fundamental diagram. Note, that Q_{out} is close to the maximum uncritical free flow at ρ_1, but exceeds it.

Schönhof and Helbing [127] distinguish five different basic types of congestion originating at an on-ramp. A schematic drawing of these types is given in Figure 2.2. The types are called *Oscillating Congested Traffic* (OCT, Figure 2.2a), *Stop-and-Go Wave* (SGW, Figure 2.2b), *Homogeneous Congested Traffic* (HCT, Figure 2.2c), *Pinned Localized Cluster* (PLC, Figure 2.2d), and *Moving Localized Cluster* (MLC, Figure 2.2e).

[1] A wide jam is a long jam as seen from the side.

Which type emerges depends on the flow upstream of the on-ramp and the merging ramp flow in relation to the flows of the fundamental diagram at densities ρ_1 to ρ_4. In the following we denote the free flow at density ρ_1 with Q_1.

If the sum of the upstream flow and the ramp flow exceeds the maximum congested flow Q_{out} a kind of extended congestion may develop (OCT, SGW, or HCT). Else, if the sum is less than Q_{out} but larger than Q_1, an emerged congestion cannot grow (PLC, MLC).

A localized cluster becomes a moving one (MLC) if the upstream flow itself is larger than Q_1. Otherwise, the localized cluster is pinned (PLC) at the on-ramp, because it requires upstream flow and ramp flow to exceed the minimum critical flow Q_1.

In case of extended congestion (OCT, SGW, or HCT) the concrete type depends on the density within the congestion. If it exceeds ρ_4 the congestion is stable, i.e., no perturbation can change it, thus it is always HCT. Between ρ_3 and ρ_4 it may be HCT, but a large enough perturbation may trigger oscillations (OCT). Below ρ_3 even small perturbations cause oscillations, thus it is OCT or—in case of lower densities—its sub-type SGW. Contrary to pure OCT, SGW has sections of free traffic between the oscillations.

We conclude that traffic jam resolution requires to make sure that the upstream flow into the jam is less than the maximum congested flow Q_{out} at the downstream front of the jam.

Kerner and Rehborn [79] identified a third traffic phase beside free and congested traffic, which they call *synchronized traffic*. In a synchronized state vehicles on all lanes move with similar velocity, but slower than free traffic and faster than in congestion. Different than free traffic, there is no fixed relation between density and flow, in fact there are state transitions in all directions in the fundamental diagram, although Neubert et al. [109] have shown that some synchronized states exhibit a certain cross-correlation between density and flow. Based on this three-phase traffic theory Kerner and Klenov [73], Kerner [70], and Kerner and Klenov [75] derive different congestion patterns than described above. Kerner et al. [77] found additional structures within the jam phase in the context of the three-phase traffic theory. However, the existence of synchronized traffic as third traffic phase is still under discussion.

Schönhof and Helbing [127] mentioned that synchronized states form a dense area close to the free-traffic branch in the fundamental diagram. Kim and Keller [83] confirm this empirically.

Cassidy and Mauch [16] have shown and empirically confirmed that along a road with several pairs of off- and on-ramps of which each pair has a net inflow, density decreases and flow increases.

Treiber and Helbing [137] explain that speed limits stabilize traffic, because they slow down fast vehicles, thus reducing velocity differences and making perturbations more unlikely. Descending slopes have the same effect. Ascending slopes, on

the other hand, amplify the speed differences, thus destabilize traffic.

Neubert et al. [109] have discovered that the time headway (temporal distance to the predecessor) distribution of free traffic has a major peak at 0.8 s and a minor peak at 1.8 s. In synchronized traffic the peak at the larger time headway becomes major and in congested traffic only the peak at 1.8 s remains.

2.3. Lane changing

There are not many empirical investigations of lane changing behavior and its impact on traffic flow. However, an observation made at Californian carpool lanes illustrates that it does have an impact on flow [65]. Carpool lanes are special lanes usually only permitted to vehicles occupied with several people. To support environmentally friendly cars, low-emission vehicles used to be allowed on Californian carpool lanes as well. In 2011 this permission got revoked, which had the unexpected effect that traffic flow on all lanes including the carpool lanes dropped, although only a few percent of the vehicles were banned. The increased density on the regular lanes led to more congestion and that also slowed down the now less occupied carpool lanes as vehicles changed between regular and carpool lanes.

A recent larger empirical investigation of lane changing behavior was published by Lee, Olsen, and Wierwille [95]. They observed test drivers on real roads using vehicles equipped with sensors to log the distances to the surrounding vehicles, and with cameras and eye-trackers to observe the test drivers. However, the study focused on the drivers' behavior and did not investigate the effects on traffic flow.

3. Traffic Models

Traffic research has a long history of building mathematical models of traffic behavior to study and describe traffic phenomena. In this chapter, we explain different approaches to traffic modeling and give some examples of traffic models, which had and still have major impact on traffic research. More details can be found in Chowdhury, Santen, and Schadschneider [17], Helbing et al. [52], Hoogendoorn and Bovy [59].

3.1. Approaches

Traffic models are separated into three classes of detail: macroscopic, microscopic, and submicroscopic models. *Macroscopic models* describe traffic by speed and density as functions of position and time without considering individual vehicles. They are similar to physical many-particle-system models. *Microscopic models* describe the behavior of individual vehicles as they react to other vehicles in the neighborhood. *Submicroscopic models* include modeling the behavior of certain vehicle parts like the steering or the brakes. The purpose of submicroscopic models differs from the goal of this work, so we do not consider them.

As an example of a macroscopic model, we present the model by Lighthill and Whitham [98], which was independently found by Richards [119], thus it is called *Lighthill-Whitham-Richards* model in the literature. It is one of the oldest models that continues to impact traffic research, as it is derived from natural assumptions, which are part of every model. It is based on a continuity equation of flow and density deduced from the fact that vehicles cannot appear or disappear:

$$\frac{\partial \rho(x,t)}{\partial t} + \frac{\partial Q(x,t)}{\partial x} = 0 \quad . \tag{3.1}$$

Here ρ and Q are density and flow as continuous functions of position, x, and time, t. Lighthill, Whitham, and Richards assume the equilibrium flow Q_e to depend directly on the density (*equilibrium flow* is the flow corresponding to the given density):

$$Q(x,t) = Q_\mathrm{e}(\rho(x,t)) \quad . \tag{3.2}$$

Inserting this into Equation (3.1) yields a wave equation with a propagation velocity $C(\rho)$:

$$C(\rho) = \frac{dQ_\mathrm{e}}{d\rho} \quad . \tag{3.3}$$

This explains why the gradient of the jam branch of the fundamental diagram (Section 2.1) corresponds to the velocity of jam fronts.

Many more recent macroscopic models are extensions to the Lighthill-Whitham-Richards model. Others follow different approaches borrowed from many-particle physics.

This thesis is about traffic improvement strategies implemented in individual vehicles. To test and prove them we need models at the level of individual vehicles, so from now on we only look at microscopic models. For an overview over models of all types (including microscopic) see Shvetsov [130].

Microscopic models are also called *car-following models*, because the vehicle directly ahead of the vehicle currently being considered (often called *our vehicle*) is often the only vehicle the behavior of our vehicle depends on, at least it is usually the most important one. The vehicle directly ahead is called *predecessor*. We give variables referencing to the predecessor usually the index "pred".

Car-following models are usually time-discrete and could be position-discrete as well. The time- and position-discrete models are known as *cellular automaton traffic models*. Cellular automata avoid floating point arithmetic, which previously had a bad impact on computation performance. On today's computers this is of less concern.

Microscopic models can be compared to macroscopic models by averaging over neighboring vehicles.

An additional class of models are *mesoscopic models*, which are a hybrid of microscopic and macroscopic models. Mesoscopic models simulate individual vehicles like microscopic models, but their behavior does not depend directly on the neighboring vehicles. Instead the vehicles' behavior is determined by the local average of certain parameters like in macroscopic models. Mesoscopic models allow to derive microscopic parameters from macroscopic values.

Car-following models are further distinguished into *one-*, *two-*, and *three-phase models* [148]. One-phase models have only stable solutions of their model equations, hence the system always has a stable equilibrium depending on the boundary conditions. A two-phase model has some unstable solutions, such that it can exhibit a free traffic phase and a congested phase. Both phases can coexist under certain conditions, thus two-phase models exhibit three traffic states. Jost and Nagel [67] explain in detail the difference between one- and two-phase models. Three-phase models have for some boundary conditions many solutions, leading to a third, synchronized phase.

As a model becomes more detailed, the number of vehicles and/or driving time that can be simulated in a reasonable computation time decreases. Hence, it is crucial for investigating a traffic phenomenon by simulation to select a model which is detailed enough to exhibit the phenomenon in question but not too detailed to make simulation infeasible.

3.2. Important car-following models

3.2.1. Wiedemann model

The Wiedemann model is an early car-following model developed by Wiedemann [160] [see also 60]. It is, for example, implemented in the well-known commercial traffic simulation tool VISSIM, created by the company PTV[1].

Depending on the gap and the velocity difference to the predecessor, the model distinguishes different types of reactions. If the gap is large enough, there is no reaction to the predecessor. When the gap drops below a perception threshold, which is larger with higher velocity difference, the vehicle enters the reaction regime, where it starts to decelerate. From there it may enter the unconscious reaction regime, where the velocity difference oscillates to keep a certain gap. Additionally, there is a deceleration regime in which the vehicle has to brake to keep a desired minimum gap and a collision regime in which this is no more possible.

The Wiedemann model is sometimes called a psychophysical model, because it considers the driver's mental state regarding the predecessor, that is if the driver reacts consciously to the predecessor or not and how that is done.

3.2.2. Nagel-Schreckenberg model

The Nagel-Schreckenberg model [108] is a cellular automaton model, which had major impact on traffic research, because despite its simple update rules it exhibits realistic traffic behavior including spontaneous traffic jams. Its simplicity made it possible, for the first time, to do research on traffic congestion by simple computer simulation instead of intricate observation of real traffic. More than 2300 citations of the original publication, up to now, illustrate its significance.

The model consists of a one-dimensional array of cells. Each may contain one vehicle with an assigned integer velocity v between zero and $v_{\max}$ set to 5 in the original publication. An update is done in these steps:

1. If the velocity is less than $v_{\max}$ and the gap to the predecessor larger than $v+1$ the velocity is increased by 1. Else, if the gap to the predecessor is less or equal to v the velocity is decreased by 1.
2. For all vehicles with non-zero velocity, the velocity is decreased by 1 with a given probability.
3. Each vehicle is moved v cells.

The integer character of the cellular automaton together with the integer velocity enabled fast simulation with the computers available around 1992 giving the opportunity to study traffic without time-consuming observation of real roads.

[1] `http://www.ptv-vision.com`

Since the publication of the Nagel-Schreckenberg model many extensions and variants have been invented and investigated by many researchers including the original authors. An example for this is the slow-to-start rule, which introduces an asymmetry between acceleration and deceleration [122]. Another example is the Comfortable Driving Model (CDM), that mimics the tendency of the driver to overreact on brake lights of the vehicle in front [84]. Schneider and Ebersbach [126] modify the Nagel-Schreckenberg model to consider the velocity of the predecessor as well.

3.2.3. Totally Asymmetric Exclusion Process

A Totally Asymmetric Exclusion Process (TASEP) is a one-dimensional cellular automaton with an asymmetric update rule [118, 121]. Each cell can be occupied with a vehicle or not. During the update the vehicle moves to the next cell with a certain probability p unless that is occupied as well. The boundary conditions are given by two more probabilities. On each update step a vehicle enters the first cell if it is empty with an entry probability p_e and, at the other end, if the last cell is occupied a vehicle leaves with another probability p_l.

Evans, Rajewsky, and Speer [28] have found an analytical solution to a TASEP with simultaneous update rules.

3.2.4. Gipps model

The design goals of the Gipps model [40] were to develop a model, which not only mimics real traffic, but also has parameters with physical meanings such that they can be easily obtained from empirical data. Additionally, an update time step Δt on the order of the reaction time τ should be sufficient.

The Gipps model computes the velocity of the next time step as minimum of a free traffic velocity v_{free} and a safe velocity v_{safe} to avoid collision with the predecessor:

$$v_{\text{next}} = \min[v_{\text{free}}, v_{\text{safe}}] \quad . \tag{3.4}$$

The free traffic velocity v_{free} is computed as

$$v_{\text{free}} = v + 2.5\, a\tau \left(1 - \frac{v}{v_{\max}}\right) \sqrt{0.025 + \frac{v}{v_{\max}}} \tag{3.5}$$

with the current velocity v, the maximum acceleration a, the reaction time τ, and the maximum velocity $v_{\max}$. This expression reflects that the engine's torque rises with increasing speed and falls back to zero when approaching $v_{\max}$. The constants are the result of measuring the acceleration behavior of real cars.

The safe velocity v_{safe} is computed as

$$v_{\text{safe}} = -b\tau + \sqrt{(b\tau)^2 + 2bg - b\tau v + \frac{b}{b_{\text{pred}}} v_{\text{pred}}^2} \tag{3.6}$$

with the maximum brake deceleration b of our vehicle and b_{pred} of the predecessor, the gap g to the predecessor, and its speed v_{pred}. For consistency with the other models we denote the maximum decelerations b and b_{pred} as absolute values contrary to the original publication of Gipps.

The safe velocity expression is derived by considering the extreme scenario that the predecessor brakes with b_{pred} until stop and our vehicle does the same after a reaction time delay during which it keeps its current velocity. More precisely, Gipps allows an additional delay θ to introduce a safety reaction time $\tau + \theta$ and shows that $\theta = \tau/2$ is sufficient to avoid collisions in the long run. This finally leads to Equation (3.6).

3.2.5. Krauß model

The Krauß model is used extensively in this work, so we present it in more detail than the other models. It was developed by Krauß [88].

The Krauß model itself

The Krauß model is explicitly based on the Gipps model by using the same approach to distinguish a preferred velocity of the driver and a safe velocity to avoid collision with the predecessor.

The driver's preferred velocity is modeled as the maximum velocity. In terms of reality, the road's speed limit, additional bad weather limits, the car's maximum speed, or the highest speed a driver feels comfortable with may be applied as maximum velocity.

As with the Gipps model the collision avoidance is modeled as a safe velocity v_{safe} depending on the velocity of the preceding vehicle v_{pred} and the gap g to it. Additionally, the safe velocity is constructed such that it obeys a maximum deceleration b and a limited reaction time τ of the driver until he/she starts braking.

With a given velocity-dependent minimal braking distance $d(v)$ Krauß assumed collision freeness if the following holds for the vehicle's velocity v:

$$d(v) + v\tau \leq d(v_{\text{pred}}) + g \quad . \tag{3.7}$$

Thus, if the predecessor tries to stop as quickly as possible and our vehicle starts to brake as hard as possible after the reaction time τ both vehicles come to stop bumper-to-bumper.

With a Taylor approximation around the mean velocity $\bar{v} = (v + v_{\text{pred}})/2$ the safety condition is derived to be

$$v \leq v_{\text{pred}} + \frac{g - v_{\text{pred}}\tau}{\frac{\bar{v}}{b} + \tau} \quad . \tag{3.8}$$

Applying the safety condition to discrete simulation steps of length Δt yields a safe velocity v_{safe} at time $t + \Delta t$ of

$$v_{\text{safe}}(t + \Delta t) = v_{\text{pred}}(t) + \frac{g(t) - v_{\text{pred}}(t)\tau}{\frac{\bar{v}(t)}{b} + \tau} \quad . \tag{3.9}$$

Together with the maximum velocity $v_{\max}$ and a limited maximum acceleration a, this limits the desired velocity v_{des} of the vehicle to

$$v_{\text{des}} = \min\left[v_{\max},\ v(t) + a\Delta t,\ v_{\text{safe}}(t + \Delta t)\right] \quad . \tag{3.10}$$

To model congestion and traffic jams every model needs randomness. Krauß solved this by introducing a random deceleration applied to the desired velocity to finally set the velocity of the next simulation step

$$v(t + \Delta t) = \max\left[0,\ \text{rand}(v_{\text{des}} - \varepsilon a\Delta t,\ v_{\text{des}})\right] \quad . \tag{3.11}$$

$\text{rand}(x, y)$ is a function which returns an equally distributed value between its arguments. ε is a model parameter between 0 and 1 to configure the actual randomness of the model. Krauß usually sets it to $\varepsilon = 1$. See also Jost and Nagel [67] for an investigation of the meaning of different values of ε.

Because of the equal distribution, the expected value of the random deceleration is $\varepsilon a\Delta t/2$. It is important that the random deceleration never exceeds $a\Delta t$, in other words that $\varepsilon \leq 1$, because that is the maximum velocity gain during the next simulation step (Equation (3.10)). A larger random deceleration produces an expectation value of the velocity close to zero. It is not exactly zero, because the random deceleration has a lower bound of zero (Equation (3.11)).

Finally, the position x of the vehicle is updated by applying the new velocity to the current position:

$$x(t + \Delta t) = x(t) + v(t + \Delta t)\Delta t \quad . \tag{3.12}$$

Krauß proved that the model fulfills the safety condition Equation (3.7) provided $\Delta t \leq \tau$ holds.

In addition, Krauß showed that the model with the dimensionless parameter values $a = 0.2$, $b = 0.6$, and $\rho = 0.3$ (density) generates realistic traffic jams. Dimensionless means setting $\tau = \Delta t = \ell = 1$ (ℓ is the vehicle length). Using the dimensions $\tau = \Delta t = 1\,\text{s}$ and $\ell = 7.5\,\text{m}$, as suggested by Krauß, the dimensionful

Table 3.1.: Parameters of the Krauß model and their values reproducing realistic traffic jams tested by Krauß

Symbol	Parameter	Value
a	Maximum acceleration	$1.5\,\mathrm{m/s^2}$
b	Maximum deceleration	$4.5\,\mathrm{m/s^2}$
Δt	Time step length	$1\,\mathrm{s}$
τ	Reaction time	$1\,\mathrm{s}$
$v_{\max}$	Maximum velocity	$\gg a\Delta t$
ε	Randomness	1

equivalents are $a = 1.5\,\mathrm{m/s^2}$, $b = 4.5\,\mathrm{m/s^2}$, and $\rho = 40\,\mathrm{veh/km}$. In addition, the maximum velocity $v_{\max}$ has to be large compared to the maximum acceleration in a simulation step. Table 3.1 summarizes the parameters of the Krauß model and their realistic values, as we use these values for most applications of the Krauß model in this work.

Krauß also mentioned, that a vehicle might have to decelerate with up to $v_{\max}/\Delta t$ within a simulation step because of the v_{safe} limit, but a deceleration above the maximum b happens rarely in practice. We have checked this in our simulations and can confirm that the effect is negligible. It does not only happen rarely, but also the observed deceleration in such cases is usually not much larger than the configured maximum deceleration b.

Further thoughts on the Krauß model

The safe velocity v_{safe} (Equation (3.9)) does not depend on the simulation step length Δt. So, shorter simulation steps make a vehicle decelerate quicker when approaching a slower predecessor instead of performing the same deceleration in larger steps.

On the other hand the maximum random deceleration (Equation (3.11)) does depend on the simulation step length Δt. Thus, the shorter the simulation steps are, the less randomness exhibits the model. This lowers the probability of a perturbation large enough to cause congestion.

Both effects make it infeasible for our purpose to choose a $\Delta t < \tau$ without changing the behavior of the model. Thus, in our simulations we always set $\Delta t = \tau = 1\,\mathrm{s}$.

Another thought is a derivation of the gradient of the jam branch in the fundamental diagram. In a stationary traffic jam the velocity is determined by v_{safe}. Suppose all vehicles are in equilibrium then all velocities are the same. Inserting v for all velocities in Equation (3.9) yields the gap $g = v\tau$. Together with the vehicle length ℓ (including the minimum gap between standing vehicles) the density of the

jam is

$$\rho = \frac{1}{g + \ell} = \frac{1}{v\tau + \ell} \quad . \tag{3.13}$$

As all vehicles move with the same velocity v this is the average velocity $v = Q/\rho$. Together with the latter equation we get the flow-density relation of the jam branch

$$Q = \rho v = \frac{1}{\tau} - \rho\frac{\ell}{\tau} \quad . \tag{3.14}$$

Thus, the slope is

$$\frac{dQ}{d\rho} = -\frac{\ell}{\tau} \quad . \tag{3.15}$$

This is also the speed at which the upstream end of a queue of vehicles, moving at equilibrium velocity v_{jam}, grows by incoming vehicles: Each time-headway interval τ one vehicle approaches the queue and prolongs it by ℓ plus the equilibrium gap of the queue, $v_{\text{jam}}\tau$. During the same time the queue moves downstream by a distance of $v_{\text{jam}}\tau$, thus gap and movement cancel out such that the queue grows upstream with a velocity of $-\ell/\tau$.

3.2.6. Intelligent Driver Model

The Intelligent Driver Model (IDM) was developed by Treiber, Hennecke, and Helbing [139] to exhibit all types of empirically observed congestion without intrinsic randomness. Randomness could be inserted by the initial vehicle distribution or by randomly inserting vehicles into an open system.

The model equations compute the acceleration of a vehicle as function of its current velocity v, desired velocity v_0, desired minimum gap s^* and actual gap s to the predecessor (a is the maximum acceleration as before):

$$\dot{v} = a\left[1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s}\right)^2\right] \quad . \tag{3.16}$$

The desired minimum gap s^* itself depends on the vehicle's current velocity v and the velocity difference Δv to the predecessor ($\Delta v > 0$ if approaching):

$$s^*(v, \Delta v) = s_0 + s_1\sqrt{\frac{v}{v_0}} + Tv + \frac{v\Delta v}{2\sqrt{ab}} \quad . \tag{3.17}$$

The concrete behavior depends on seven parameters listed in Table 3.2. In general all parameters except δ can be vehicle-dependent.

The first two summands in Equation (3.16) model the acceleration on a free road and the last summand models deceleration when approaching a predecessor. When approaching a standing vehicle the latter produces a small deceleration as

Table 3.2.: Parameters of the IDM

Symbol	Parameter
a	Maximum acceleration
v_0	Desired velocity
δ	Acceleration exponent
s_0, s_1	Jam gaps
T	Safe time headway
b	Desired deceleration

long as the gap is large, a deceleration around b when coming closer and becoming small again when our vehicles becomes slower. This is why the authors called the model "intelligent driver".

On a free road an acceleration exponent $\delta = 1$ produces exponential relaxation to v_0. An acceleration exponent $\delta \to \infty$ produces a constant acceleration with a. The authors suggest to set $\delta = 4$.

The jam gap s_0 is the bumper-to-bumper gap between standing vehicles. The jam gap s_1 is usually set to zero to keep the model simple, but it is needed for some special cases.[2] See Treiber, Hennecke, and Helbing [139] for details. The authors show that the time headway T is the most interesting parameter to generate different traffic phenomena. A time headway is the time a vehicle needs to advance through the gap at its current velocity.

An extension of the IDM is the Human Driver Model (HDM), which adds delayed anticipation of the vehicle's own velocity, velocity difference to the predecessor, and gap to the predecessor to the IDM [140]. The HDM is a meta-model that could be applied to other car-following models as well.

3.2.7. Optimal Velocity Model

Another widely-used model to investigate traffic congestion is the Optimal Velocity Model (OVM). It was published by Bando et al. [4].

The principle of the OVM is that the driver's acceleration is proportional to the difference between the gap-dependent optimal velocity $v_{\mathrm{OV}}(g)$ and the current velocity v

$$\dot{v} = s\left[v_{\mathrm{OV}}(g) - v\right] \quad , \tag{3.18}$$

where s is a constant that encodes the sensitivity of the driver. It is assumed to be the same for all vehicles. The optimal velocity function v_{OV} must be monotonically increasing with an upper bound of $v_{\max}$.

[2] More recent publications applying the IDM omit the summand with the factor s_1 completely, see for example Kesting, Treiber, and Helbing [82].

Bando et al. showed that a closed system with a fixed number of vehicles is linearly stable if the derivative of v_{OV} at the equilibrium gap g_{e} is larger than half of the sensitivity:

$$v'_{\mathrm{OV}}(g_{\mathrm{e}}) > s/2 \quad . \tag{3.19}$$

The authors additionally did non-linear stability analysis by numerical investigation. This requires a concrete optimal velocity function v_{OV}. After testing and discarding a too simple function they suggest:

$$v_{\mathrm{OV}} = \tanh(g - g_0) + \tanh g_0 \quad . \tag{3.20}$$

This optimal velocity function accelerates the vehicle if the current gap g is larger than the desired gap g_0 and vice versa, which produces realistic traffic jams.

Helbing and Tilch [53] found that the OVM produces unrealistically high accelerations and is not collision free if a vehicle approaches a standing vehicle from a large distance.

Wilson et al. [161] found that on a circular road shock waves of stop-and-go traffic steepen up to crashes if the sensitivity s is too low, but a larger sensitivity would result in unrealistically large accelerations. After investigating other solutions they suggest a multiple-look-ahead optimal velocity function to solve these problems.

Since the original publication, many other optimal velocity functions were proposed and investigated, including functions that additionally depend on the velocity of the predecessor. The OVM is especially popular for mathematical analysis, because it exhibits a transition from free traffic to congestion within a single, continuous expression. See Batista and Twrdy [5], Chuan-Yao et al. [19], Dong, Weng, and Li [24], Helbing and Moussaid [54], Lenz, Wagner, and Sollacher [96], Ye-Liu et al. [99], Orosz, Wilson, and Krauskopf [111], Peng and Sun [114], Peng [115], Peng and Sun [116], Tang et al. [134], Tutiya and Kanai [144], Wei, Ning-Guo, and Yu [157], Wen-Xing and Lei [159], Yu, Shi, and Zhou [172].

Opposed to the Krauß model and the IDM, in the OVM the maximum acceleration and deceleration are the same by absolute value for all optimal velocity functions, but many traffic researches assume that the asymmetry between acceleration and deceleration is important for realistic congestion development.

3.2.8. Kerner-Klenov model

The Kerner-Klenov model [72] (the appendix of Davis [23] presents a more comprehensible overview of the model equations) is designed to exhibit first-order transitions between free, synchronized, and jammed phases. Thus, it is a three-phase model. The model has nine free parameters, which makes it rather complex. Like the Krauß model, it is a stochastic model, but it applies several different probabilities depending on the state of the predecessor. Interestingly, it incorporates

v_{safe} from Krauß, Wagner, and Gawron [89], which presents a precursor of the Krauß model.

The main feature of the Kerner-Klenov model is that a driver in synchronized state accepts an arbitrary gap to the predecessor between a maximum gap and a safety minimum gap, while keeping the same velocity as the predecessor. This produces a two-dimensional region in the fundamental diagram.

Variations of the Kerner-Klenov model are presented by Kerner and Klenov [74], as well.

3.2.9. Comparisons

Brockfeld and Wagner [12] and Brockfeld, Kühne, and Wagner [11] have compared more than ten different car-following models, including most of the models presented here, according to their ability to reproduce real traffic data. The data are from different sources including double-loop detectors on a highway and data from dedicated driving experiments on a test track with different strict driving instructions. The general result is, that all simulators are equally good at simulating real traffic data.

Wagner and Nagel [148] compared one-, two- and three-phase models by their ability to reproduce certain traffic patterns in an artificial scenario mimicking a temporary road block followed by a temporary flow limit at the obstacle. The result is a clear difference between one- and two-phase models, but less difference between two- and three-phase models. One-phase models are not able to reproduce empirical congestion, but it could not be decided if two- or three-phase models are better for this purpose.

3.3. Lane change models

All microscopic traffic models describe the longitudinal behavior of vehicles. However, when modeling traffic jams on highways with microscopic models, lane changing is also important, because many congestion phenomena are caused or at least driven by lane changes (Section 2.3).

Furthermore, we are going to investigate an active driving behavior strategy including scenarios in which not all vehicles comply. For those scenarios passing of vehicles could be important.

Microscopic lane-change models usually have two aspects: incentive and safety. An incentive to change the lane can be the wish to pass slower vehicles, the necessity to continue the route, or simply random. When the incentive is sufficient to prefer another lane, it must be checked if it is safe to change the lane. The safety check usually considers the current states (position, velocity) of the vehicles ahead and behind the gap into which the current vehicle intends to change.

3.3.1. Krauß' lane change model

In the traffic simulator CircSim, that we developed for this work (Section 6.3), we implemented a lane change model suggested by Krauß [88]. Therefore, we explain it in more detail.

Lane changes depend on v_{safe} and on $v_{\text{safe, dest}}$. The latter is v_{safe} under the assumption the vehicle would already be on the destination lane at its current longitudinal position. The lane change model distinguishes congested and non-congested situations. Congestion is assumed if both v_{safe} and $v_{\text{safe, dest}}$ are below a threshold v_{cong} (set to 60 km/h by Krauß). Now, it is preferred to change

to the left lane if it is not congested *and* $v_{\text{safe}} < v_{\max}$.

to the right lane if $v_{\text{safe}} \geq v_{\max}$ *and* $v_{\text{safe, dest}} \geq v_{\max}$.

Krauß is ambiguous in handling the cases in which v_{safe} or $v_{\text{safe, dest}}$ is equal to $v_{\max}$, but this is negligible for numeric floating point numbers, because in numerical computations equality of floating points should always be tested with limited accuracy.

Under congestion a lane change is not determined, because it has no significant advantage and passing on the right is allowed (see below). Note, that the preference of the right lane in above rules contradicts congestion, as well.

If neither change is preferable a change is claimed nevertheless with a certain small probability, even under congestion.

The lane change is considered safe if the safety criterion for longitudinal movement is fulfilled for all vehicles after the supposed change. In detail the following conditions have to be met for all surrounding vehicles:

- $g \geq v_{\text{pred}}$
- $v_{\text{safe}} \geq v - b\Delta t$

The subtraction of $b\Delta t$ is done to ensure the safe velocity for the next simulation step.

Finally, there are some rules to prevent passing on the right unless we have congestion. If not congested, vehicles have to obey $v_{\text{safe, left}}$ as well as their own v_{safe}, except $g_{\text{left}} < v_{\text{pred, left}}\Delta t$, where the index "left" denotes safe velocity, gap, and v_{pred} with respect to the predecessor on the left lane. The exception is necessary, because of the discrete simulation steps: Without the exception a vehicle on the left lane which just passed our vehicle may produce a very small g_{left} for one simulation step inducing a very small $v_{\text{safe, left}}$, which would make our vehicle brake hard. This is entirely unrealistic.

3.3.2. Other lane change models

An example for a rather complicated lane change model is the urban lane change model by Gipps [41]. It distinguishes between a preferred lane the driver needs to take to follow his/her route and a target lane the driver eventually changes to for other reasons. The model defines many decisions to find the appropriate target lane depending on the distance to the next turn, speed advantage, avoiding to follow a heavy vehicle, avoiding an obstruction, use an HOV (Heavily Occupied Vehicles) lane if allowed to, or even use an HOV lane exceptionally to pass an obstruction. The closer the vehicle is to a turning point the heavier braking and smaller gaps are accepted to get into the destination lane. The model uses the Gipps model (Section 3.2.4) as longitudinal traffic model, but other car-following models may easily fit into it.

Treiber and Helbing [138] have extended their IDM by the lane change model MOBIL (Minimizing Overall Braking Induced by Lane-Changes). The safety criterion of MOBIL allows to change the lane when the supposed lane change does not require the follower on the destination lane to decelerate more than a certain threshold.

A lane change is preferred in MOBIL if the sum of all accelerations of the affected vehicles would be raised by more than another threshold. Hence the name Minimizing Overall Braking Induced by Lane-Changes. In the sum, the accelerations of the other vehicles are weighted with a politeness factor to model the aggressiveness of the driver. An additional weighting of the accelerations can model the prohibition to pass on the right lane and/or the avoidance of the right lane to prevent to be stuck behind a truck to apply European habits of lane usage.

Some multiple-lane models only consider the correlations between neighboring lanes without actually performing lane changes [62, 133]. Instead they are like longitudinal car-following models that additionally consider the distance and velocity difference to the predecessor on the neighboring lane with a certain weight. This should model explicitly traffic on Chinese roads, where drivers always have to be prepared for sudden non-indicated lane changes.

4. Existing Traffic Control

Assisting road traffic control by technical devices has a long history. In 1868 the first traffic signal adopted from earlier railroad semaphores was installed at a crossing in London [105]. It displayed at daylight by arms and at night by gas-powered red or green lights the signals "stop" and "caution".[1] This traffic semaphore raised a lot of interest in continental Europe and the United States and attracted many visitors. Up to the 1920s many different types of traffic signals got developed and installed until the growing number of motor vehicles made standardization necessary.

This chapter describes modern approaches to traffic control on highways, which are supposed to improve the traffic flow. If not otherwise noted the research results presented in this chapter stem from the publications by Hegyi, Bellemans, and De Schutter [49] and Papageorgiou et al. [112].

4.1. Traffic measurement

To control the development of traffic situations, its past and current states have to be measured. In Chapter 2 we already described video cameras and induction loops. Induction loops are the main method of gathering traffic data for the strategies presented in the following.

Induction loops cannot directly measure traffic density, because density is a spatial value and induction loops are located at a single position. As a replacement, traffic engineers use the *occupancy* of an induction loop, which is the fraction of time the induction loop is occupied by a vehicle. Note, that the occupancy is only an estimation of the density; depending on the distribution of velocities and vehicle lengths it might deviate from the actual density.

Besides those, some cities experiment with floating car data. A fleet of taxis, buses, or trucks is equipped with GPS devices which regularly send their current position to a special server. A large example of this is the real-time traffic information of Google Maps using GPS data from registered GPS-enabled cell phones.[2] On-board navigational devices with online information systems also receive congestion information based on cell phones [125]. An example of a local system applying

[1] Different than today's traffic lights, its purpose was not vehicular traffic control but pedestrian safety.

[2] `http://googleblog.blogspot.de/2009/08/bright-side-of-sitting-in-traffic.html`

taxi-fleet data beside classical traffic measurement is the VAMOS system of the city of Dresden, Germany.[3]

A recent publication suggests to apply car-to-car communication to detect traffic jam waves and distribute this information upstream [61].

4.2. Ramp metering

Ramp metering is the most investigated approach to improve traffic behavior. It controls the inflow into a highway at an on-ramp using a special traffic light. Some systems allow only a single vehicle to pass during one green phase, whereas others allow as many vehicles to pass as possible, like regular traffic lights at intersections. The latter approach is called *bulk metering* and controls the inflow by adapting the relative lengths of the green and red phases as required. Ramp metering has usually a defined minimum inflow to prevent backlogs from growing too much.

Ramp metering has two modes of operation. *Traffic spreading mode* lets the vehicles pass at the same rate they arrive at the on-ramp, but spreads the demand evenly over time to avoid large perturbations. Particularly, spreading mode breaks arriving platoons to make merging smoother. Ramp metering in *traffic restricting mode* holds back vehicles if too many arrive. Its purposes are to prevent breakdowns on the highway by keeping the density below a critical value and to influence the route choice of drivers. In addition, it could be applied to localize shock waves at the on-ramp instead of their normal propagation upstream. Prevention of breakdowns and localizing jam waves have the positive side-effect to prevent blocking of off-ramps which are upstream of on-ramps by congestion.

Ramp-metering strategies are classified in *fixed-time* and *traffic-responsive* strategies. Fixed-time strategies run at a predetermined rate depending on the time of day, which is derived from historical traffic data. Traffic responsive strategies require some kind of permanent traffic measurement from which the current metering rate is derived.

As an example of a traffic-responsive ramp metering strategy, we present the often implemented *demand-capacity* metering strategy. The metered ramp flow, Q_{ramp}, depends on the occupancy, o_{down}, downstream of the ramp (where ramp and highway flow are merged) and the current flow on the highway approaching the on-ramp, Q_{inflow}. Parameters of the strategy are the highway's maximum flow, Q_{cap}, a critical downstream occupancy, o_{crit}, and the ramp's minimal flow, $Q_{\text{ramp, min}}$. The metering rate at time step t is determined as

$$Q_{\text{ramp}}(t) = \begin{cases} Q_{\text{cap}} - Q_{\text{inflow}}(t-1) & \text{if } o_{\text{down}}(t-1) \leq o_{\text{crit}} \\ Q_{\text{ramp, min}} & \text{otherwise.} \end{cases} \tag{4.1}$$

[3] `http://www.vamosportal.de`

The demand-capacity strategy is a feed-forward control, thus it may drift away from a pre-specified value and it is very sensitive to the correct choice of parameters. A feedback control would be to make the ramp flow directly proportional to the difference between o_{down} and a critical or desired downstream occupancy.

Prevention of traffic breakdowns by ramp metering avoids congestion at the price of not utilizing the full capacity of the highway. Kerner has proposed a strategy which avoids this [69]. It allows congestion to occur and, if it happens, lowers the ramp's inflow to quickly resolve the traffic jam. Congestion is detected by measuring the average upstream velocity and comparing it to a pre-defined threshold.

Zhang and Levinson [173] present results of an experiment in a metropolitan area to determine the efficiency of ramp metering. For a period of seven weeks all ramp metering in the area was completely turned off. Traffic measurements of these seven weeks were compared to the corresponding seven weeks of the previous year with the usual ramp metering. Depending on the type of ramp-metering-controlled bottlenecks they found up to 10 % increased capacity by ramp metering.

According to Hegyi, Bellemans, and De Schutter [49], ramp metering improves travel times between 0.39 % and 30 %. Treiber and Helbing [137] have shown that optimal metering results in much shorter travel times, even when the additional waiting time at the on-ramp is included; however, deviating a little from the optimal metering makes this advantage decrease quickly.

4.3. Variable speed-limits

Variable speed-limits are speed-limit signs which change the announced limit automatically, depending on continuously measured traffic data. Usually, longer sections of heavily utilized highways are equipped with variable speed-limit installations every few kilometers, which is also known as *corridor control system*. Their main purpose is to increase safety by reducing the number of accidents on crowded highways [123].

Schick [123] has compared the maximum flow on sections with corridor control systems to similar sections without such systems. He found that variable speed-limits on their own increase the highway capacity only slightly. Geistefeldt [39] has confirmed this: On observed two-lane highways the average capacity without any speed limit[4] was 3842 vehicles/km, with a permanent limit of 100 km/h or 120 km/h it was 3943 vehicles/km, and with a variable limit it was 3950 vehicles/km. However, variable speed-limits considerably reduce the probability of traffic collapses.

Hegyi, De Schutter, and Hellendoorn [49, 50] explain two approaches of variable speed-limit strategies: homogenization of traffic by issuing speed limits above the

[4]The survey was done on German highways, which have no default speed limit.

critical speed (the speed at the point $(\rho_2, Q_{\max})$ in the fundamental diagram in Figure 2.1), and prevention of traffic breakdowns by lower speed limits, which may resolve existing congestion. According to Hegyi, De Schutter, and Hellendoorn [50] the effect of homogenization on highway performance is negligible and for breakdown prevention no field observation results are available.

However, Hegyi, De Schutter, and Hellendoorn [50] present an integrated control system that coordinates ramp metering and variable speed-limits of a road network. It applies ramp metering to prevent traffic breakdowns and, if this is not sufficient because of the minimum metering rate, variable speed-limits help to accommodate the ramp's additional flow. As mentioned above, a minimum metering rate is required to prevent too large backlogs into the secondary road network.

Beyond these, integrated traffic-control systems apply *model-predictive control*, a common approach in control engineering. Here, the current and recent states of the system are used to feed a simulation of the system, running until a certain time horizon. Then the predictions of the simulation are applied to the actuators, i.e., ramp metering and variable speed-limits. This is repeated for every time step of the controls.

Hegyi, De Schutter, and Hellendoorn [50] also investigated their model-predictive approach with main-stream metering instead of variable speed-limits. Main-stream metering limits the inflow at the start of the highway like ramp metering limits it at on-ramps. They have found a reduction of up to 14.3 % of the total time vehicles spent in the network when applying ramp metering and variable speed-limits and up to 17.4 % when applying ramp and main-stream metering.

Carlson, Papamichail, and Papageorgiou [14, 15] suggest and investigate feedback-controlled variable speed-limits to prevent the activation of highway bottlenecks. Without control, congestion appears if the inflow to a bottleneck is larger than its capacity (known as *bottleneck activation*). The suggested controller strategies move the congestion a few 500 m upstream of the bottleneck, such that the flow through the congestion is higher than the flow through an activated bottleneck would be and less than the bottleneck's critical flow. According to simulations, the total time of the vehicles spent in the network can be reduced by up to 19.8 %.

Another interesting approach is Colorado's rolling speed harmonization on the I-70.[5] Instead of variable traffic signs, the variable speed-limit is enforced by police cars leaving from a start point every 5 to 10 minutes at a certain speed depending on the current traffic situation. No vehicle is allowed to overtake such a police car, thus it forms a uniform platoon moving at constant velocity. The efficiency is evaluated on a daily basis to improve the strategy, resulting in increased safety and traffic flow.

[5] `http://www.coloradodot.info/news/2012-news-releases/03-2012/i-70-speed-harmonization-sunday`

4.4. Routing

Avoiding or resolving congestion can be done by getting drivers to select different routes to their destinations if the planned routes or the roads they are currently on are congested.

Papageorgiou et al. [112] distinguish *pretrip* and *en route* congestion information. Pretrip information may be obtained by internet, phone service, television or radio broadcasting and may lead to postponing the planned trip, choosing a different departure time, a different mode of transportation, or a different route. Chrobok et al. [18], Marinósson et al. [102], Mazur et al. [103] have implemented a model-predictive congestion forecast system delivering pretrip information. The results can be accessed by a website, which recommends the fastest route to reach a given destination at a given time. It can even alarm the driver in time to reach the destination at the desired time.

En route information may be received by radio, in-car equipment, or variable message signs, which are electronic traffic signs displaying arbitrary messages to the drivers. En route information should trigger different routing decisions at oncoming bifurcation points.

There are two possible types of route guidance information on a variable message sign. Either it displays the current state of the next road section, or it displays alternative route recommendations together with the estimated travel times on the road ahead and on the alternative route. There is usually not enough space to display both types of information. Drivers often prefer to know the current state, but using this information requires road network knowledge which has to be recalled within a few seconds after noticing the variable message sign. Instead, displaying alternative routes gives the operators more influence on the traffic demand on different roads. However, Papageorgiou et al. [112] mention that only a few strategies have been tested in the field so far.

Route guidance can be optimized towards system or user optimum with respect to the travel times. A system optimum minimizes the total costs of all drivers, while a user optimum minimizes individual costs, such that no driver can reduce his or her individual costs by choosing an alternative route.

Variable message signs are also applied to display “keep your lane” directives [49]. This is useful at critical flow, at which velocities on all lanes are similar. Keeping the lane reduces perturbations which otherwise may trigger congestion.

Davis [23] investigated a simple localized rerouting strategy. If the average velocity at a 500 m long section on the right lane near an on-ramp is below a threshold, which equals the maximum velocity on the related previous off-ramp, some vehicles are rerouted to take the off-ramp. The fraction of rerouted vehicles depends on the ratio between the average velocity and the threshold velocity. This strategy is able to prevent congestion at the on-ramp, but it does not care about the further path of the rerouted vehicles.

An alternative to pretrip or en route congestion information is tolling of road sections or areas. Varaiya [147] investigates different combinations of tolling bottlenecks or single lanes and ramp metering to reduce congestion. A combination of ramp metering and tolling bottlenecks is most effective.

Braess [7, 8] has shown that adding more links to a road network to reduce the traffic load of the remaining network might be counterproductive. Under certain conditions an additional link reduces the network throughput provided each driver takes its individual best route. This is known as the *Braess paradox*. Yang and Bell [170] present a workaround for the Braess paradox.

4.5. Special lanes

Traffic flow may also be improved by using specified lanes differently than the remaining ones. Schick [123] reports that temporary unblocking of hard shoulders during peak hours considerably increases the flow because of the increased road capacity, but that this approach has impact on the safety. Hegyi, Bellemans, and De Schutter [49] note that unblocking of the hard shoulder is only helpful if the downstream road can accommodate the additional traffic flow. Another approach is to assign center lanes of arterial roads to different directions depending on the current demand, which is known as *tidal flow*.

The city of Atlanta, Georgia, USA has implemented a dynamic tolling system called Peach Pass.[6] Paying the toll gives access to express lanes on equipped highways. The toll rate is dynamically adapted to the current traffic demand to ensure free flow on the express lanes. Using any other lane is free of charge.

4.6. Adaptive Cruise Control

Different than all previously presented control systems, *Adaptive Cruise Control* (ACC) is a vehicle-local system. An ACC device continuously measures the distance to the vehicle ahead and automatically controls the speed to keep a given temporal distance (time headway). For some years now, such systems are commercially available.

First proposals of radar systems to automatically keep a certain distance at high speeds were made in the 1960s [97] with the objective to increase speed and density on highways. Peppard [117] found that measuring the distance to the vehicle ahead is not sufficient for stable speed control, which he called *string instability*. To overcome this, at least the distance to the successive vehicle has to be measured as well.

[6] `http://www.peachpass.com`

Vahidi and Eskandarian [145] give an overview of applications of ACC. It was published before ACC was available but when it had been already announced by the car manufacturers. ACC as a driver assistance system needs to keep a temporary distance larger than the distance an individual driver would normally keep, as otherwise the driver would feel uncomfortable.[7] In addition, if the driver has to take over control in an emergency response situation he/she needs extra time to react. This additional distance reduces the maximum density and thus the capacity of the road. Furthermore, Vahidi and Eskandarian [145] mention investigations which have shown that drivers are less likely to use ACC in heavy traffic. Instead, they prefer to turn it on in fog, at night, when driving for a long time, or in low density traffic.

Kesting et al. [80, 81] present a strategy to improve the entire traffic flow by ACC. The ACC information and the vehicle's own velocity is used to determine the current traffic situation and switch to an appropriate driving behavior. The detected traffic situations are free traffic, approaching congestion from upstream, moving in congested traffic, leaving congestion downstream, and passing infrastructural bottlenecks. The behavior is adapted by changing the comfortable acceleration, deceleration and temporal distance. Simulations have shown that 25 % equipment rate is sufficient to avoid congestion and at 5 % equipment rate, mitigation of congestion can be observed.

Hegyi, Bellemans, and De Schutter [49] report about several research programs in the 1990s to increase the highway capacity by making vehicles travel automatically in dense platoons with a gap of a few meters between the individual vehicles and larger gaps between the platoons. To achieve this, still many safety, legal, and also psychological problems need to be solved, hence such systems can not be expected in the near future. However, current research tries to find control strategies to make such platoons stable and robust [46, 47, 48, 104]. Almost all of the proposed control strategies require a local car-to-car communication network, which passes information in both directions within the platoon.

Other than the usual ACC strategy to keep a fixed temporal distance, Kerner [71] suggests an ACC-based control algorithm, which he calls Driver-Alike-ACC (DA-ACC). As long as the spatial gap to the predecessor falls between a safe minimum gap and a predefined maximum gap the exact distance is of no concern, and the velocity is kept the same as the velocity of the predecessor. Kerner reports that if 30 % of the vehicles are equipped with DA-ACC, under certain simulated conditions, free flow exists where otherwise synchronized flow is observed.

[7] For example, the driver's manual of the Volkswagen Golf 6 (dated 2009-03-06) explains that the driver can set the temporal distance in 5 steps between 1 s and 3.6 s.

4.7. Car-to-car communication

There are many scientific projects applying wireless communication among vehicles (car-to-car or C2C) and between vehicles and other entities (C2X, this may include other vehicles) for many different purposes including the ones described above, although none of the systems is commercially available yet.

Many projects set up and test C2X with real cars and roadside infrastructure. These projects are usually split into establishing the communication itself (hardware and lower protocol layers) and developing applications of the communication. The most common applications are emergency warnings and real-time congestion information. The primary use case for emergency warnings is an accident behind a curve, which oncoming vehicles might recognize too late without C2X. Examples of C2X projects are:

FleetNet Additional to the already mentioned applications FleetNet has the goal to provide internet access in cars [33, 37].

SOTIS (Self-Organizing Traffic Information System) SOTIS is a sub-project of FleetNet to develop protocols for pure C2C without road-side components. Beside other information it disseminates en route congestion information [164, 165, 166]. According to simulations an equipment rate of 2 % is sufficient to disseminate individual information over more than 50 km [167].

CVIS (Cooperative Vehicle-Infrastructure Systems) is an open system which also addresses privacy and security questions.[8]

NoW (Network on Wheels) is a successor of FleetNet [34] which develops different data dissemination protocols for different purposes and also addresses privacy and security concerns.

simTD is a testbed for C2X applications with real cars on real roads. The project permanently employs 20 professional drivers and up to 480 additional ones. The professional drivers are trained to establish predetermined traffic situations in which the behavior of the additional drivers is observed while they use a C2X assistance system to be tested.[9]

C2C-CC (Car-2-Car Communication Consortium) is an association of European car manufacturers and research institutions to provide a C2X infrastructure and develop applications with the goal to get it on the market [2].

ETSI ITS Standardization activities of the European Telecommunications Standards Institute (ETSI) regarding Intelligent Transportation Systems (ITS).[10]

[8] `http://www.cvisproject.org`
[9] `http://www.simtd.de`
[10] `http://www.etsi.org/technologies-clusters/technologies/intelligent-transport`

> The goal of ETSI ITS standards is to cover all modes of transportation (automotive, railways, aeronautical, and maritime) and all types applications. These standards are based on the results of the previous and other C2X projects.

Beside these large C2X projects, many research results are published on single aspects of C2X infrastructure and applications. They are mainly based on simulations and sometimes based on mathematical analysis. We mention here some publications which are related to traffic flow and congestion.

Cottingham and Davies [21] present a vision of the development of C2C- and C2X-based communication networks and their applications. Regarding congestion they suggest that vehicles send their movement data into the network to get back aggregated congestion information to adapt their routes. An earlier paper of Cottingham, Davies, and Beresford [22] briefly describes an urban traffic network with fixed access points to collect and disseminate congestion information. This system reduces the average journey time by up to 6 % according to simulations.

Tondl, Jobmann, and Meincke [136] describe an algorithm to disseminate through pure C2C emergency warnings caused by an accident. The goal of the algorithm is to evaluate the relevance of forwarding an emergency message to avoid overloading the transmission capacity with unnecessary repetition of the message. According to simulations, above a certain vehicle density the number of transmitted messages is nearly independent of the number of vehicles.

Eichler et al. [27] investigated a pure C2C strategy to make vehicles avoid a road which is blocked by a traffic breakdown. They measured by simulations the average velocity depending on the density of vehicles in the city and their equipment rate. Only at the highest tested equipment rates and densities the strategy decreases the average velocity, in all other cases the strategy improves the average velocity. The decrease at high vehicle densities comes from too many vehicles bypassing the blocked road, which causes congestion on other roads.

Kerner, Klenov, and Brakemeier [76] tested a strategy to prevent or postpone a general congested pattern (GP) at a bottleneck. A GP in Kerner's three-phase traffic theory is similar to HCT in Helbing's classification (see Section 2.2). The strategy detects when synchronized flow occurs at the bottleneck and makes the vehicles upstream of the bottleneck keeping larger gaps to their predecessors as a consequence. In a simulated test case applying the Kerner-Klenov model this results in a widened synchronized flow moving with 60 km/h as opposed to 40 km/h without the strategy.

Schönhof et al. [128] and Kesting, Treiber, and Helbing [82] simulated a strategy propagating information about traffic jam fronts in upstream direction by applying vehicles moving in the opposite direction as data carriers. This works reliably if at least 3 % of the vehicles are equipped with C2C.

An approach to reduce congestion by C2C was presented by Lee and Kim [94]

(also available as preprint [93]). It is based on a cellular automaton model published by Lee et al. [92] in which each vehicle drives either optimistic or defensive. Optimistic driving occurs if the predecessor moves faster than the driver's own vehicle and the pre-predecessor faster than the predecessor, or when the pre-predecessor moves at nearly maximum velocity. In optimistic driving a vehicle keeps a shorter safety gap to the predecessor than in defensive driving, because in optimistic driving the driver does not expect the predecessor to decelerate soon. The C2C strategy of Lee and Kim [93] makes additionally a vehicle switch to defensive driving if the mth vehicle ahead moves slower than a given threshold v_{th}. By simulation of a single-lane circular road they found $v_{\text{th}} \approx 43\,\text{km/h}$ and $10 \leq m \leq 30$ to work best. With growing vehicle density it keeps up a larger synchronized flow for longer than without C2C, but at higher densities the flow with the C2C strategy is worse than without.

Knorr and Schreckenberg [85] and Knorr et al. [84] suggest another C2C strategy to reduce congestion. When the average velocity of the equipped vehicles ahead falls below a threshold the following equipped vehicles increase their gap to the predecessor by the reaction distance, i.e., the distance traveled within the reaction time. The authors report a travel time decrease of 15 % if 40 % of the vehicles are equipped. The model contains a random deceleration like the Krauß model—although it is implemented differently—and the additional gap is achieved by reducing the random deceleration when it exceeds the reaction distance. However, in our simulations we found that a reduction of the random deceleration leads to less congestion anyway (see Section 7.1).

Wang, Kulik, and Ramamohanarao [150] present an on-ramp merging strategy based on C2C. Before reaching the merging point vehicles on the ramp and the main road decide about the merging order and adapt their velocities accordingly. This increases the flow on a ring road, but it assumes that all vehicles are equipped with C2C.

5. AutoNomos Concepts

This work was part of the DFG project AutoNomos [32, 152]. The goal of AutoNomos was to improve traffic organization and flow by ad-hoc networks between vehicles [55].

The AutoNomos members Sándor P. Fekete, Stefan Fischer, Horst Hellbrück, Christiane Schmidt, and Axel Wegener developed several concepts into which the results of this work are embedded. Those are described in this chapter.

5.1. HDC, OIC, and ADS

Hovering Data Clouds (HDCs), Organic Information Complexes (OICs), and Advanced Distributed Strategies (ADSs) are the central concepts of AutoNomos.

These concepts are based on the observation that some phenomena emerge from correlated behavior of many entities without being bound to a certain set of entities. Contrary, the entities carrying a phenomenon may become replaced continually by further entities taking their place. The phenomenon depends on the presence of sufficiently many entities but not on individual entities.

To model such phenomena AutoNomos developed the concept of *Hovering Data Clouds* (HDCs) [25, 156]. An HDC is a data structure describing such a phenomenon. HDCs are hosted by the entities currently forming the phenomenon. They are automatically passed to entities entering the phenomenon and usually removed from entities leaving it. An HDC is created if some neighboring entities discover—by local communication—a correlation among themselves forming a possible phenomenon. HDCs are disseminated through the network such that all possibly interested entities know about them.

Some emergent structures consist of several locally emerging phenomena. Their AutoNomos counterpart is the *Organic Information Complex* (OICs). HDC-hosting entities form OICs by finding matching HDCs in their repositories of received and self-generated HDCs. With OICs the entities derive global patterns of emergent structures.

Finally, *Advanced Distributed Strategies* (ADSs) apply the global knowledge contained in the HDCs and OICs to improve the global situation for the benefit of all entities.

We now demonstrate these concepts with a traffic jam as example of an emerging collective phenomenon and the vehicles both as entities forming the phenomenon

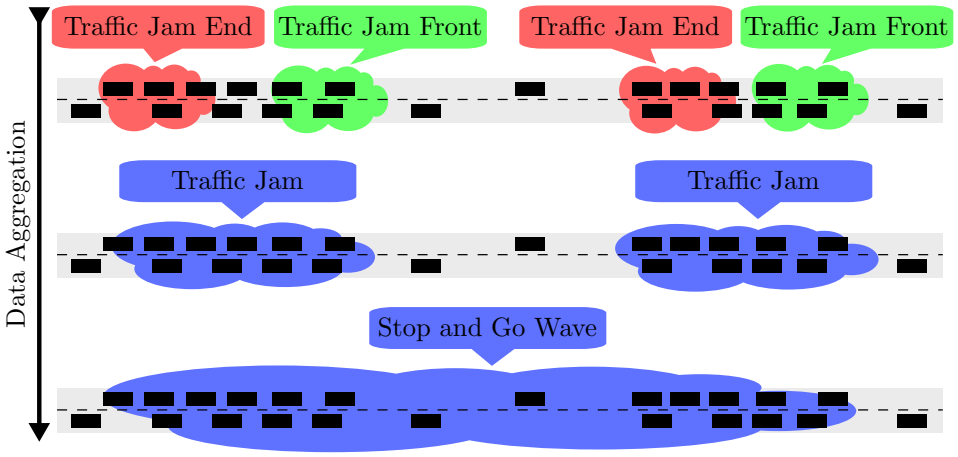


Figure 5.1.: HDCs and OICs. (Diagram from Ebers et al. [25])

and as hosts for HDCs and OICs. The road at the top of Figure 5.1 shows how HDCs rise at the upstream and downstream ends of congestion by vehicles detecting collective deceleration or acceleration in their neighborhood, respectively. The HDCs remain at the congestion ends and fronts, although the vehicles pass on through the congestion. See Fekete et al. [31] for an algorithm to generate HDCs at traffic jam fronts.

The HDCs are passed to the other vehicles on the road. This way all vehicles come to know about start- and endpoints of congestion and aggregate these information to OICs as the road in the middle of Figure 5.1 illustrates. The bottom road of Figure 5.1 shows how HDCs may be aggregated to more elaborate OICs containing more detailed patterns of congestion.

5.2. Architecture

Figure 5.2 shows the architecture of the implementation of HDCs, OICs, and ADSs developed by project AutoNomos [25]. A vehicle featuring an implementation of this architecture is called an *equipped vehicle* and the fraction of equipped vehicles is called *equipment rate* or *penetration*. The implementation is the same in each vehicle. It has three external interfaces. On the left is the interface to local sensor readings, such as GPS position, current velocity, and current acceleration. Some modern vehicles are equipped with ACC (Section 4.6), that permanently measures the distance to the predecessor. These data may be of interest as sensor reading,

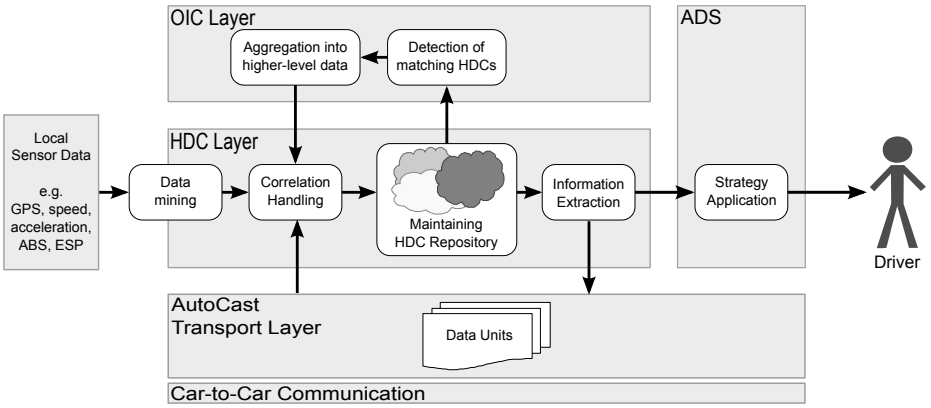


Figure 5.2.: AutoNomos architecture. (Variation of a diagram from Ebers et al. [25])

as well.

At the bottom is the interface to a wireless radio device to set up the ad-hoc network among the vehicles. It applies a special protocol named *AutoCast*, which is explained in the next section.

Finally, on the right we have the interface to give recommendations to the driver to execute the strategies depending on current HDC and OIC data.

The HDC layer at the center is responsible for exchanging data with all other components. It parses the data from the sensor readings and correlates them with the data received from other vehicles and the OIC information. The result is stored as HDCs in a repository. From the HDCs, it extracts relevant information and passes it to AutoCast for sharing with other vehicles and to the ADSs. Furthermore, the HDC layer provides the HDC repository to the OIC layer.

The OIC layer scans the HDC repository to detect HDCs with information belonging to the same traffic phenomenon and aggregates those to form an OIC data structure. Finally, the ADS component uses the HDC information to compute recommendations for the driver.

This architecture is implemented in every vehicle, which sets up a distributed feedback loop among the vehicles: The loop between the HDC layer and the AutoCast layer is connected via communication to other vehicles.

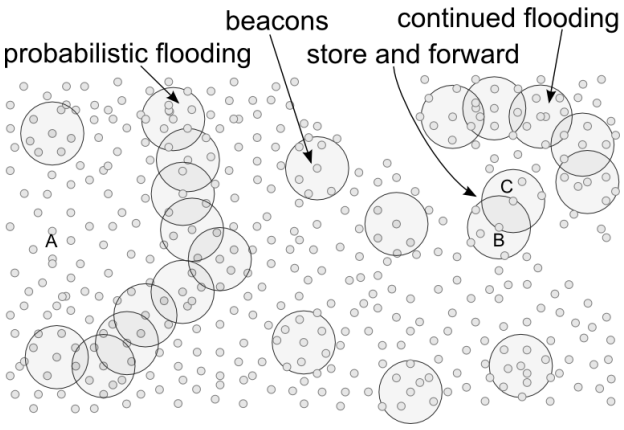


Figure 5.3.: Data dissemination in AutoCast at different densities of hosts in an arbitrary environment. (Diagram from Ebers et al. [25])

5.2.1. AutoCast

AutoCast is a wireless communication protocol developed in project AutoNomos for car-to-car communication. Its core feature is to cover the entire vehicle density range from sparse free traffic to dense traffic jams. In sparse traffic it takes care not to lose information and in dense traffic it takes care not to overrun the radio bandwidth with the same data all over. All AutoCast communication is based on broadcasts, thus it does not need to maintain lists of communication partners. We give a rough overview of AutoCast here, see Ebers et al. [26], Hellbrück, Wegener, and Fischer [56], Wegener [151], Wegener et al. [153] for details.

AutoCast achieves this by joining different protocols depending on the local density of network hosts. The local density is derived by periodic beacon messages of the hosts, so each host always knows the number of single-hop neighbors. See Figure 5.3 for an illustration of the following explanations.

The first technique is to not always broadcast all data itself. Instead periodic beacons contain only hashes (or other IDs) of the data units known to the sender beside some, but not all, complete data units. A receiver of the beacon message compares the received list of IDs with its own list of IDs to determine which data units are missing in its own repository and which data units are missing in the senders repository. Then the receiver adds the latter as complete data units to its subsequent beacon messages. Thus, hosts implicitly request missing data units. This technique ensures quick dissemination of data units without retransmitting already known data units all over.

Another technique is probabilistic flooding with the probability of sending mes-

sages depending on the local network host density by a special function found empirically by simulation experiments.

Finally, store and forward helps to join partitioned networks. If a vehicle has no neighbors to share data units with it stores the data unit even if its geographical bounds are violated, in the prospect of meeting other vehicles interested in the data while moving along the road. If that happens probabilistic flooding takes place again. With store and forward, vehicles going in the opposite direction help to maintain HDCs.

5.3. Jam-ADS

This work mainly concentrates on developing an ADS to improve the flow of highway traffic by avoiding congestion. To express this, we call it *Jam-ADS*. Jam-ADS anticipates the behavior of vehicles ahead and derives velocity recommendations to avoid or at least reduce congestion on the road, such that traffic flow increases and—as a side effect—fuel consumption becomes less [29].

Jam-ADS can also be seen as a system to extend the driver's view ahead beyond the direct predecessors. As it would not benefit the driver to present him/her in detail positions and velocities of many vehicles ahead, the data needs to be aggregated and transformed to useful information and recommendations.

Jam-ADS is not designed to affect the vehicle directly by overriding the driver's control. Giving driving recommendations instead, avoids safety issues and keeps the driver's authority over the vehicle.

5.3.1. Objectives

From the general goals of Jam-ADS we derive the following concrete objectives:

1. It should depend entirely on local communication among the vehicles.
2. It should prevent traffic jams and congestion.
3. It should resolve existing traffic jams.
4. It should reduce fuel consumption.
5. It should work with limited equipment rates.
6. It should be self-organizing.
7. It should increase the traffic capacity (maximum flow) of the road.
8. It should increase traffic safety or at least keep the current safety level.

Objective 1 means Jam-ADS should get along without a central facility to aggregate and analyze the data from the vehicles or any other external infrastructure. It should suffice to implement Jam-ADS in enough vehicles to see its effects.

We need objective 5 to ease market introduction, because we cannot expect that all vehicles will be equipped at once. To get customers to pay for Jam-ADS they must have a good perspective to benefit from it. Furthermore, there will probably always be non-equipped vehicles left or at least some drivers not complying to the recommendations of Jam-ADS.

Objective 6 refers to the definition of self-organization and adaptivity according to Herrmann, Werner, and Mühl [57]. A self-organizing system is a system that has organizational features similar to some biological systems like, for example, ant colonies. They consist of a larger number of identical (or very similar) components that interact with each other without central control, such that the system maintains a stable structure and adapts well to external changes even under loss of some components. See Fromm and Zapf [38] for a research overview and discussion of self-organization and Wolf and Holvoet [168] for an explanation of the difference between emergence and self-organization. Objective 6 kind of summarizes some of the other objectives.

Objective 7 refers to modern theory of network flows [35, 86, 131] that is applied to optimize traffic flow through road networks as well. Classical network flow theory presumes a fixed capacity (maximum flow) on each arc of the network, i.e., as long as the flow is below the capacity the velocity on the arc is constant. More recent research has investigated load-dependent transit times, i.e., the more flow an arc has to carry the slower are the vehicles. But, all these network flow theories assume a soft or hard capacity limit given by macroscopic relations between flow and average velocity like an unbreakable physical law. Our objective 7 is to go beyond this by increasing the capacity itself by changing the microscopic behavior of the vehicles.

Objective 8 is self-evident, because the introduction of new technologies is generally not accepted by individuals and society if it reduces the current safety level.[1]

5.3.2. How are the objectives achieved?

According to traffic research [51, 68] traffic can be in a stable, metastable or instable state. In an instable state, all density perturbations lead to traffic jams. In metastable traffic, perturbations need to be above a certain threshold to cause a traffic jam (see Section 2.2). Thus, to prevent traffic jams the strategy needs to avoid perturbations or at least reduce too large perturbations in metastable traffic.

[1] This objective appears also in legal requirements, for example, about conversion of a railway system to driverless vehicles.

Traffic jams have a typical outflow, that is lower than the maximum free flow (capacity) of the road. To resolve a traffic jam the strategy needs to reduce the upstream inflow into the jam such that it is less than the jam's outflow at the downstream end until the jam vanishes.

Fuel consumption can be split into different energy consuming effects (see Section 6.3.1) generating different parts of the total consumption. An important part is caused by acceleration, particularly in dense traffic where vehicles often have to decelerate and reaccelerate. To reduce fuel consumption the strategy should reduce these deceleration/acceleration cycles.

5.3.3. Basic algorithm

This section describes the Jam-ADS algorithm we mainly investigated. To distinguish it from other variants we call it *average-strategy* where necessary. The idea of this algorithm comes from Christiane Schmidt and Christopher Tessars.

The basic idea is to recommend a velocity $v_{\text{Jam-ADS}}$ which is a convex combination of the driver's desired velocity v_{des} and the average velocity of the vehicles ahead v_{avg}:

$$v_{\text{Jam-ADS}} = \lambda\, v_{\text{des}} + (1 - \lambda)\, v_{\text{avg}} \quad . \tag{5.1}$$

λ is a parameter between 0 and 1 which determines the weighting of the input velocities. Section 9.2 proves that recommending v_{avg} directly (corresponding to $\lambda = 0$) is counterproductive.

The convex combination leads to the question which is the best λ and how to compute v_{avg}? λ may depend in general on other parameters; v_{avg} may be taken over a certain number of vehicles ahead, or within a certain distance.

Furthermore, we avoid to recommend a velocity larger than v_{des} to avoid safety issues, because we assume that the driver desires to go as fast as safely possible when following another vehicle:

$$v_{\text{rec}} = \min\left[v_{\text{des}},\ v_{\text{Jam-ADS}}\right] \quad . \tag{5.2}$$

This also ensures a safe velocity for car-following simulations.

Another problem is how to discover v_{des} of a real driver? In case of simulations we know it as the velocity coming out of simulation before applying Jam-ADS and eventually randomness. But, we cannot expect a real driver to permanently tell the system his/her intended velocity and then drive according to the recommended velocity of our system. As a solution we can measure the velocity of the predecessor and distance to the predecessor with ACC (see Section 4.6), apply the Krauß model or another car-following model to these data, and use its result as v_{des}.

The algorithm has the additional benefit to increase safety by reducing the speed of upstream vehicles when approaching a traffic jam, which is often the cause of fatalities.

6. Traffic Simulation

Many results presented in this thesis are gained by traffic simulation. This chapter describes the applied simulator software. SUMO and Shawn (Section 6.2) are coupled open-source traffic and communication simulators that we extended to meet the needs of this work. CircSim (Section 6.3) is a combined network and traffic simulator we developed for special simulation tasks of this work.

6.1. Technical terms

We use some special terms regarding simulations presented in this thesis that we define here.

A *vehicle-state* contains the position (longitudinal and lane) and velocity of a vehicle at a certain simulation step. The set of all vehicle states at a simulation step is called a *micro-state*.

We call *model-parameter* a parameter that is part of a simulation configuration and whose value determines the simulation itself. An individual model-parameter may not have an influence on a specific simulation if it is not applicable because of other model-parameter settings; for example, changing λ or the Jam-ADS distance has no influence if Jam-ADS is globally turned off by another model-parameter to select the strategy to apply. Repeating a simulation run with the same values of the model-parameters reproduces it. Not all configuration parameters are model-parameters in this sense, because some configuration parameters do not have an influence on the simulation result at all, for example parameters regarding debug output or automatic e-mail notifications.

Simulations executed collectively to test different model-parameter values are called a *simulation session*. A simulation session is usually executed by a control script that automatically loops over different values of model-parameters. See Section 6.4 for details about our control scripts.

The set of model-parameters varied in a simulation session is called *model-parameter set* (of the session). A single simulation within a simulation session belonging to specific values of the model-parameter set is called a *(single) simulation run*. In other words, different simulation runs of a simulation session differ in at least one model-parameter value.

The random seed applied to simulation runs is also a model-parameter, because many simulation sessions contain simulation runs that differ only in their ran-

dom seed to exclude occasional effects. The random-seed model-parameter can be set to either generate a new random seed on each run or to a fixed seed. If a simulation run with a fixed seed is repeated it reproduces each time the exact same micro-states, provided the same simulation software versions are applied (see Section 6.4.6).

6.2. SUMO, Shawn, and TraCI

SUMO is a traffic simulator and *Shawn* a communication network simulator and both are coupled over a TCP/IP connection running the *TraCI* protocol defined for this purpose. All of these are implemented in C++ [63, 64].

6.2.1. SUMO

SUMO[1] (Simulator of Urban MObility) is an open-source traffic simulator developed and maintained by the Institute of Transportation Systems at the German Aerospace Center[2] (DLR) [6, 87]. It is released under the *GNU General Public License* [43] and contains contributions of many volunteers.[3] Despite its name it can also simulate highway traffic.

According to Joerer, Sommer, and Dressler [66] in recent years (2009–2011) more than 20 % of the publications regarding C2X communication, applied SUMO for road traffic simulation, hence we assume SUMO to be a valid vehicular traffic simulation tool.[4]

SUMO can run simulations on arbitrary road networks consisting of edges and junctions. Edges may have several lanes. At the junctions the lanes of different edges are individually connected. Networks could be imported from public sources (for example Open Street Map[5]), from other traffic simulators (for example MATSim[6]), generated by SUMO's tool `netgen`[7], or set up manually. To set up a road network manually, the user has at least to define junctions (nodes) by their coordinates and the streets connecting the junctions (edges). A special tool called `netconvert` converts these input data into a network configuration file for SUMO.

For SUMO each vehicle goes exactly one trip, i.e., it starts at a certain time and follows a specified route of successive edges. During simulation, SUMO decides

[1] Available at `http://sumo.sourceforge.net/`
[2] `http://www.dlr.de/fs/en/`
[3] For example, Flötteröd [36] has developed a framework to calibrate traffic simulators and implemented it for SUMO.
[4] Some projects using SUMO are listed at `http://sumo.sourceforge.net/doc/current/docs/userdoc/Other/Projects.html`
[5] `http://www.openstreetmap.org/`
[6] `http://matsim.org/`
[7] `netgen` can generate random and regular networks (grid, spiderweb).

dynamically which lane the vehicle takes.

SUMO offers several tools to generate traffic demand and convert it to a set of vehicle trips. Trips could be generated completely random, by turning probabilities for each junction, by *origin-destination matrices*[8], by flows, or completely manually. It is also possible to dynamically assign new routes to vehicles on the road by defining so-called *rerouters* or by TraCI commands (Section 6.2.3).

To make configuration easier, vehicles usually belong to a vehicle type which defines the properties of the vehicles belonging to it. Basic properties are, among others, maximum velocity, maximum acceleration, maximum deceleration, and vehicle length. In addition, a car-following model is assigned to a vehicle type, which may have special properties. SUMO's default car-following model is a variant of the Krauß model (Section 3.2.5).

SUMO inserts vehicles into the first edges of their individual routes. There are several vehicle attributes to configure vehicle insertion such that SUMO can more or less freely chose the actual insertion position, lane, and velocity. If insertion at the start time of the trip is not possible under the given insertion constraints without violating the safety conditions of the car-following model, insertion is postponed until it becomes possible. This may cause a growing backlog of vehicles waiting to be inserted as soon as possible. Thus, depending on the particular configuration, there is a natural maximal flow SUMO can generate on an insertion edge.

SUMO detects intersecting vehicles. In case of such a collision the rear vehicle is "teleported". Teleportation is a mechanism in SUMO to resolve collisions and other difficult situations. During teleportation a vehicle is removed from the road and moved along its route with the average velocity of the edge it left, until it can be reinserted into an edge.

We observed in our simulations that collisions happened exceptionally often during initial insertion into the first edge of the vehicle's route, causing teleportation to a further downstream road section. To avoid reinsertion of such a teleported vehicle into a road section we want to evaluate, we increased the number of edges provided for insertion. See Section 6.4 for details.

v_{safe} in SUMO

SUMO applies the Krauß model with a slightly different computation of v_{safe}. Originally Krauß proposed the expression (Equation (3.9))

$$v_{\text{safe}} = v_{\text{pred}} + \frac{g - v_{\text{pred}}\tau}{\frac{\bar{v}}{b} + \tau}$$

[8] In traffic research origin-destination (OD) matrices are lists of transportation demands between different zones at different times of day.

for v_{safe}, but SUMO uses

$$v_{\text{safe}} = -b\tau + \sqrt{(b\tau)^2 + v_{\text{pred}}^2 + 2bg} \quad . \tag{6.1}$$

This expression can be derived by solving Krauß' safety condition given in Equation (3.8) for v regarding the v dependency of the mean velocity $\bar{v}$. It also fulfills Krauß' general safety condition Equation (3.7).

An explanation of the expression is the following scenario: The predecessor brakes as hard as possible (with deceleration b) until stop. The follower keeps his speed for one reaction time τ and then brakes as well with deceleration b until stop. If the follower initially has the velocity v_{safe} according to Equation (6.1), then both vehicles stand bumper-to-bumper in the end.

This scenario is the same as the scenario Gipps used for the derivation of his safe velocitiy (Equation (3.6)), but the resulting expressions are a little different. The difference has two reasons: The additional safety reaction time θ (see Section 3.2.4) is not considered here and Gipps always takes into account the average velocities of the current and the next time step, but Equation (6.1) only considers the current time step.

Additionally, Gipps does not assume the same maximal deceleration b for both vehicles. SUMO is able to assign a different maximal deceleration to different vehicle types, so Equation (6.1) may not always be sufficient to avoid collisions.

In fact, the version of SUMO that we applied finally in this work uses a slightly different expression for v_{safe} that considers v_{pred} in integer multiples of $b\Delta t$ with Δt as simulation step length. Let $v'_{\text{pred}} := v_{\text{pred}}/b\Delta t$ and $\lfloor v'_{\text{pred}} \rfloor$ denote the largest integer less or equal to v'_{pred}, then v_{safe} is set to

$$v_{\text{safe}} = -b\tau + \sqrt{(b\tau)^2 + 2bg + (b\Delta t)^2 \lfloor v'_{\text{pred}} \rfloor \left(2v'_{\text{pred}} - \lfloor v'_{\text{pred}} \rfloor - 1\right)} \quad . \tag{6.2}$$

For $v_{\text{pred}} \gg b\Delta t$ this expression converges to Equation (6.1) above, but for smaller v_{pred} it makes a difference in the order of $b\Delta t$.

Dynamic maximum acceleration

Opposed to the original Krauß model (Section 3.2.5) with its constant maximum acceleration a, SUMO computes the maximum acceleration of the current vehicle, $a_{\text{max, veh}}$, dynamically by the maximum acceleration of the type of the current vehicle, $a_{\text{max, type}}$, the vehicle's current velocity, v_{curr}, and the vehicle type's maximum velocity, $v_{\text{max, type}}$, by

$$a_{\text{max, veh}} = a_{\text{max, type}} \left(1 - \frac{v_{\text{curr}}}{v_{\text{max, type}}}\right) \quad . \tag{6.3}$$

Thus, if a vehicle constantly accelerates with its maximum possible acceleration, the acceleration over time approaches zero asymptotically and the velocity over time approaches the maximum velocity asymptotically, as well.

A consequence of this is the reduction of the random deceleration when approaching the vehicle's maximum velocity, because the acceleration within the random deceleration term (Equation (3.11)) has to be the current maximum acceleration. Otherwise a vehicle would lose more speed by random deceleration than it could reaccelerate in the next simulation step.

To overcome this reduction of randomness of fast vehicles, we follow a suggestion of the SUMO maintainers to configure a high maximum velocity for the vehicle type and set a lower maximum velocity of the road. This limits the overall maximum velocity, but has no influence on the maximum velocity of a specific vehicle type, thus it does not limit considerably the maximum acceleration. This approach is more realistic than the original constant maximum acceleration of the Krauß model, because for real cars the acceleration decreases continually when reaching the maximum velocity and there is also less randomness when the driver pushes the accelerator to the limit. However, at the typical small velocities of dense traffic we are interested in this does not make a considerable difference.

Lane change in SUMO

The lane change model in SUMO is complex and, unfortunately, not well documented because of permanent improvements of SUMO adapting it to more and more constraints and empirical observations.

The preferred lane of a vehicle first depends on the route of the vehicle, because neighboring lanes may be connected to different subsequent road network edges. Each vehicle has to ensure to be in due time on a lane that permits it to follow its route when approaching the end of the current edge. Furthermore, it can happen that lanes do not have a successor, like an acceleration lane of an on-ramp, for example, which also forces a vehicle to switch to another lane.

SUMO checks if a required lane change is blocked by another vehicle. Vehicles which were blocked in previous simulation steps are preferred for lane changes over vehicles which were not blocked yet. Also, vehicles which want to change the lane may get neighboring vehicles to slow down to ease the lane change. In addition, lane changing at on-ramps is treated specially.

Only if none of these cases apply, SUMO checks for preferred lanes in terms of v_{safe} by computing a lane change probability which is accumulated until the change has happened.

6.2.2. Shawn

Shawn is a communication network simulator intended to simulate wireless sensor networks, implemented in C++ [30].[9] It is developed and maintained by the Institute of Telematics of the University of Lübeck and the Algorithms Group of the Technische Universität Braunschweig and applied in many different research projects [90].

Shawn provides a framework to exchange messages between processors placed on network nodes. Processors may send messages to other processors on other nodes through Shawn's framework. The user defines a processor by deriving from a basic processor class and implementing a certain method which is called at every simulation step to perform the processor's internal computations including sending of messages. Shawn passes the messages through a configurable chain of *filters*. A filter can manipulate the message arbitrarily or even drop it completely. Shawn provides filters to simulate limited communication ranges, different types of perturbations, logging, or other purposes. In addition, the user can add self-defined filters. Processors need to implement a message-processing method to handle received messages, which is called by Shawn's framework after a message has passed the chain of filters.

Shawn also offers different node *movement models* to simulate a network of moving nodes. One of these movement models uses TraCI (see next section) to connect Shawn's nodes with SUMO's vehicles such that Shawn applies the vehicle positions to its nodes. In addition, the processors on such vehicle nodes may send TraCI commands to SUMO to change the behavior of the vehicles in future simulation steps.

Beside that, a user can define so-called *tasks* in Shawn, which are called before or after the simulation, or regularly before or after each simulation step (known as pre-step or post-step task).

A configuration file controls the execution of the simulation. It configures a world of nodes with processors, may call tasks or install them as pre- or post-step tasks, and executes the simulation for a given number of steps. Additionally, values can be assigned to parameters, which are accessible by processors and tasks.

6.2.3. TraCI

TraCI (Traffic Communication Interface) is a protocol designed to connect a traffic simulator and a communication simulator. It was originally developed by Wegener et al. [155] to connect SUMO with Shawn or other network simulators. It is now maintained together with SUMO itself. Today, several communication simulators implement TraCI to connect to SUMO.

[9] Download: `http://sourceforge.net/projects/shawn/`,
documentation: `https://www.itm.uni-luebeck.de/ShawnWiki`

TraCI is a client/server protocol build on TCP/IP with the traffic simulator as server and the communication simulator as client. Data exchange is always initiated by a command from the communication simulator and causes a single answer of the traffic simulator. It could change some SUMO-internal variables or trigger traffic simulation steps. Particularly, the traffic simulator waits for a special command to execute one or a given number of traffic simulation steps and return the new positions of the vehicles. Thus, both simulators always act alternately, although they run as independent processes of the operating system or even on different hosts.

The communication simulator can request nearly all data of the current traffic simulation state and change many of them. In addition, the communication simulator could set subscriptions of data it wants to know regularly. The traffic simulator automatically adds the subscribed data to the answer of the command to advance the traffic simulation.

There are two generations of the TraCI protocol. The first generation pragmatically provided all commands needed for a specific research project. The second generation has a more systematic structure. Except for a few control commands (mainly the command to advance the simulation), all commands are either value retrieval commands or state changing commands with the same general structure. Each command belongs to a certain type of entity (vehicle, vehicle type, edge, lane, junction, traffic light, induction loop, etc.) and first denotes the affected variable of the entity followed by the entity's ID, which is a unique arbitrary string.

Data within TraCI commands are usually preceded by a byte denoting the following data type. Data types are either atomic types (byte, unsigned byte, integer, double, string, string list[10]) or compositions of atomic types or other already composed types. Thus, each data type can be seen as a tree with atomic data types as leaves.

During our work SUMO upgraded from the first to the second TraCI generation and discontinued support for the first. Thus, we had to implement the second generation in Shawn as well, to benefit from corrections and other improvements intermediately made in SUMO. To implement the new TraCI interface as generically and flexibly as possible we implemented a hierarchy of C++ templates called TraciTypes, which can be easily plugged together to form an arbitrary data type tree. The contained values can be accessed like fundamental or struct C++ types, but TraciTypes also know how to assemble themselves to TraCI messages or parse messages to read their values. See Appendix B for details.

[10] According to the documentation, TraCI considers a *string list* as atomic data type. Presumably, because the default implementation of message en-/decoding handles it at the lower software layers.

6.2.4. Our extensions added to SUMO and Shawn

To simulate Jam-ADS with SUMO and Shawn, we had to make several extensions to both. It was not feasible to implement Jam-ADS exclusively as a special Shawn processor, because it needed some internal data of the car-following model not accessible via TraCI and it needed to be applied before the random deceleration of the Krauß model implemented in SUMO.

So we split Jam-ADS into two parts. One implemented as extension to SUMO and one as processor and special tasks in Shawn. The two parts roughly resemble a Jam-ADS device on board of a vehicle and the driver obeying Jam-ADS' recommendations. The part in Shawn is responsible to collect all relevant data from other vehicles and aggregate them. The part in SUMO is responsible to actually change the behavior of the vehicle depending on the applied strategy. This is similar to the observer/controller architecture developed by Richter et al. [120] with the Jam-ADS part in Shawn as observer and the Jam-ADS part in SUMO as controller.

Because of the splitting of Jam-ADS in Shawn and SUMO we needed to pass some internal data from Shawn to SUMO. Depending on the particular Jam-ADS strategy we passed up to two float values for each vehicle and simulation step. To do this, we extended TraCI's command to set a vehicle-state by another so-called *vehicle-variable*. The data contained in our vehicle-variable is a struct containing two doubles and an ID denoting the particular Jam-ADS strategy to apply.

The implementation of Jam-ADS in SUMO is realized as a specialization of SUMO's default traffic model described above. We overrode the method to perform the longitudinal vehicle movement to apply the particular Jam-ADS strategy. We also extended the class to receive and store the additional TraCI data of Jam-ADS.

To Shawn we added a class hierarchy of different processors implementing different Jam-ADS strategies. Furthermore, we developed two tasks. The first task (`roadmap_scan`) runs once before the simulation starts, to gain all relevant data of the road network by TraCI requests to SUMO. For the remaining simulation run, `roadmap_scan` provides these data as internal methods returning a lane's predecessors, successors, left or right neighbor lane, or length. The second task (`sim_state`) runs before each simulation step and collects current positions and velocities of all vehicles providing them to the Jam-ADS processors to avoid frequent retransmission of the same data over TraCI.

We configured a limited Jam-ADS equipment rate with the additional XML-attribute "fraction" in the vehicle type configuration. This allowed to configure the equipment rate differently for each vehicle type, for example, passenger cars and trucks. Our control scripts converted the attribute values into several flow configurations with according numbers of equipped and unequipped vehicles of each type (see Section 6.4). The equipped and non-equipped vehicle types were distinguished by different prefixes of the vehicle-ID string, so all subsequent tools

knew the equipment state without introducing an error prone additional information passing into the tool chain (Figure 6.13, explained in Section 6.4.1).

6.3. CircSim

CircSim is a traffic simulator developed by us for this work. It was used by other projects as well. Its development started with a few small scripts made by Christopher Tessars to check some hypotheses. From that we evolved it to a flexible traffic simulation tool with exchangeable traffic models and control applications. CircSim is written in MATLAB[11].

Note that Jam-ADS was initially called *AutoNomos* like the research project we worked for. Hence, CircSim still uses the term AutoNomos instead of Jam-ADS in many places.

6.3.1. Features of CircSim

This section concentrates on describing the features of CircSim from a user's perspective. Technical details for adaptation to specific needs and further development are given in Appendix A.

As the name CircSim already suggests, CircSim simulates a closed ring road with a fixed number of vehicles. Nearly everything else is configurable and could be changed by plug-in code.

CircSim was developed for first tests of Jam-ADS variants. Thus, all features are designed to be highly flexible and extendible.

Traffic models in CircSim

CircSim has a defined interface to the traffic model to apply, such that it can easily be replaced. It is even possible to replace the traffic model during an ongoing simulation run. A traffic model consists of a *drive-model* for longitudinal movement and a *lane-change-model*. The latter is further split into the *preferred-lane-model* for choosing the preferred lane and the *lane-change-safety-model* for checking if a preferred lane change would be safe. All three submodels are implemented as functions called for every vehicle at every simulation step. They have access to the complete history of micro-states.

A simulation step is done in three substeps. First, the drive-model is called for all vehicles and then the preferred-lane-model for all vehicles. The drive-model

[11] © 2012 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See `http://www.mathworks.com/trademarks` for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

may return a data structure for each vehicle, which is passed to the preferred-lane-model. This could prevent double computation of the same values, as it is the case with the default models of CircSim. Finally, for each vehicle which would like to change to another lane the lane-change-safety-model is called and on positive result the lane change is executed immediately.

The strict separation into the substeps of longitudinal driving, determining preferred lanes, and lane-change safety checks ensures that every change in velocity and lane is based on a consistent state of the surrounding vehicles. Otherwise collisions may occur if one vehicle accelerates and another vehicle switches onto its lane into the same space, because the former vehicle sees an outdated lane assignment and the latter an outdated position/velocity of the respective other vehicle.

Currently, CircSim offers the original Krauß model (Section 3.2.5) and two variants with different v_{safe}. One is v_{safe} like originally in SUMO (Equation (6.1)) and the other v_{safe} variant follows an earlier publication by Krauß, Wagner, and Gawron [89]. In the interactive dialog described below v_{safe} of the original Krauß model is denoted as "Dissertation", because it stems from Krauß' dissertation, v_{safe} according to SUMO is denoted as "SUMO", and the third v_{safe} variant as "Discrete", because it makes use of rounding of values.

Beside the Krauß model, there is an implementation of the IDM (Section 3.2.6) without a lane-change-model and not obeying traffic lights. Opposed to the original IDM, which contains no randomness the random deceleration of the Krauß model is applied to the IDM as well. But it is possible to turn off random deceleration completely for both models.

The maximum velocity of the Krauß model in CircSim is the minimum of several maximum velocity limits: the global maximum velocity, the maximum of the vehicle type, and the maximum velocity of the individual vehicle. The latter is initialized to infinity, but could be set to a finite value during a simulation run.

As lane-change-model CircSim contains Krauß' lane-change-model as described in Section 3.3.1 with a little modification: Within a traffic jam CircSim switches a vehicle already to the right lane if $v_{\text{safe, right}}$ is larger than v_{safe} on the vehicle's own lane, instead of waiting until the vehicle may move with $v_{\max}$ on both lanes, because that would prevent switching to the right lane within traffic jams. Furthermore, we extended the random lane change to more than two lanes. We first check with the configured probability if a random lane change should be done in general and then—provided the vehicle has a left and a right neighbor lane—decide with equal probability for one of the neighbor lanes. Finally, during the lane-change safety check CircSim also checks for possible collisions.

Vehicles usually take some time to switch to another lane. CircSim models this by making a switching vehicle occupy both lanes for one simulation step, such that the vehicle has a successor on the old and the new lane. Its internal lane number is the mean value of both lane numbers during that simulation step, hence in

CircSim vehicles could have non-integer lane numbers being a multiple of 1/2.

Traffic strategies are realized in CircSim as so-called *influencers*. Influencers are either *drive-influencers* or *lane-change-influencers*. Influencers are called for each vehicle immediately after the drive-model or the preferred-lane-model, respectively. They receive the result of the respective traffic model and can change it, while they have access to the complete history of micro-states and the Jam-ADS equipment property of every vehicle. Influencers of the same type (drive or lane-change) can be chained. We implemented Jam-ADS as influencers in CircSim.

After the drive-influencers the velocity is further limited by all applicable maximum velocities (road's maximum, vehicle's maximum, current velocity plus maximum acceleration) and by random deceleration. The random deceleration comes last as we do not want the influencers to limit the randomness, because the influencers should simulate a recommendation to the driver. Such recommendation cannot reduce the randomness of a human driver.

Fuel consumption model in CircSim

The fuel consumption model implemented in CircSim is taken from Wegener et al. [154] except that we do not simulate stopping the engine after some standing time.

The fuel consumption is computed independently for each vehicle and simulation step. The consumption is either regular consumption or idle consumption. The regular consumption is the sum of three parts related to acceleration, air-resistance, and rolling-resistance. The idle consumption is a minimal consumption per time to keep the engine running while the vehicle stops or is too slow for regular consumption. When averaging the consumption over several vehicles we treat the idle consumption as a fourth part beside the three regular consumption parts.

The current acceleration determines the consumption type. If the acceleration is less than a configurable fuel cut-off deceleration (set to $-0.38\,\mathrm{m/s^2}$) we assume the engine to cut off fuel supply to utilize the engine's brake capability as modern cars do, thus all consumption parts are zero. If the acceleration is between cut-off and a configurable coasting deceleration (set to $-0.18\,\mathrm{m/s^2}$), or velocity and acceleration are zero, we assume the engine is declutched, thus we have idle consumption. Otherwise, we assume regular consumption.

The three parts of regular consumption are calculated by the resistance forces the vehicle has to move against. The rolling-resistance force, f_{roll}, is computed from the rolling resistance coefficient, c_{roll}, the vehicle's mass, m, and the gravitational acceleration, g, by

$$f_{\mathrm{roll}} = c_{\mathrm{roll}}\, m\, g \quad , \tag{6.4}$$

the air-resistance force, f_{air}, from the skin friction coefficient, c_{air}, the vehicle's frontal area, A, the air density, ρ_{air}, and the current velocity, v, by

$$f_{\mathrm{air}} = \frac{1}{2} c_{\mathrm{air}}\, A\, \rho_{\mathrm{air}}\, v^2 \quad , \tag{6.5}$$

and the acceleration force, f_{acc}, from the velocity change, Δv, during the current simulation time step, Δt, by

$$f_{\text{acc}} = m \frac{\Delta v}{\Delta t} \quad . \tag{6.6}$$

CircSim converts these forces to the absolute regular fuel consumption parts, V_{roll}, V_{air}, and V_{acc}, of the current simulation step using the energy density of fuel, d, and the engine's thermal efficiency, e, by

$$V_{\text{roll, air, acc}} = \frac{f_{\text{roll, air, acc}}\, v\, \Delta t}{d\, e} \quad . \tag{6.7}$$

Note that it may happen that a small deceleration—less than coasting deceleration—results in a negative acceleration consumption, $V_{\text{acc}} < 0$. Of course, vehicles cannot have a negative fuel consumption, because that would be a conversion of kinetic energy back to fuel, but V_{acc} is only one part of the consumption, which could be negative as long as the sum is non-negative. In case of a total negative consumption, $V_{\text{roll}} + V_{\text{air}} + V_{\text{acc}} < 0$, all consumption parts, V_{roll}, V_{air}, and V_{acc}, are set to zero. Physically, a negative acceleration consumption, $V_{\text{acc}} < 0$, corresponds to the case that the accelerator pedal is released only a little, such that the energy demand of air- and rolling-resistance is partially fulfilled by fuel and partially by decreasing kinetic energy of the vehicle.

Finally, for each part and simulation step the average consumption per distance is computed as follows. Let t be the current simulation step, i a vehicle, $V_p(i,t)$, $p \in \{\text{roll, air, acc}\}$ a current absolute fuel consumption part, and $d(i,t)$ the distance traveled by vehicle c at simulation step t. Then the average fuel consumption per distance $C_p(t)$ of part p is computed as

$$C_p(t) = \frac{\sum_{i \in \{\text{vehicles}\}} V_p(i,t)}{\sum_{i \in \{\text{vehicles}\}} d(i,t)} \quad . \tag{6.8}$$

All fuel consumption parameters can be set as configuration values in CircSim. Some parameters depend on the vehicle type (skin-friction coefficient c_{air}, frontal area A, and mass m) and the remaining are global. Table 6.1 gives the fuel consumption values applied throughout this thesis for all vehicle types including trucks. Setting the same consumption values for all vehicle types makes it easier to interpret the influence of different strategies on average fuel consumption and also helps to compare the simulation results to personal experience as passenger car owner.

Simulation-state file

CircSim allows to save the entire simulation run including the complete history of all micro-states into a simulation-state file. See Appendix A.5 for a complete

Table 6.1.: Fuel consumption settings. The values below the dashed line could be set differently for each vehicle type.

Setting	Symbol	Value
Gravitational acceleration	g	$9.81\,\mathrm{m/s^2}$
Air density	ρ_{air}	$1.29\,\mathrm{kg/m^3}$
Rolling resistance coefficient	c_{roll}	0.015
Energy Density of fuel	d	$8.9\,\mathrm{kW\,h/l}$
Thermal efficiency	e	0.3
Idle consumption		$1\,\mathrm{l/h}$
Max cut-off deceleration		$-0.38\,\mathrm{m/s^2}$
Max coasting deceleration		$-0.18\,\mathrm{m/s^2}$
Skin friction coefficient	c_{air}	0.4
Frontal area	A	$2\,\mathrm{m^2}$
Mass	m	$1400\,\mathrm{kg}$

list of contained data. A simulation-state file can be loaded to analyze the data, reproduce evaluation plots (see Section 6.3.2), or resume the simulation.

Resuming a simulation can be done either interactively (see next section) or automatically by a MATLAB script. Particularly, resuming can be applied to proceed with different configurations from the same starting point.

Interactivity

Algorithms representing traffic-flow strategies usually have many parameters to change their behavior. To gain some intuition which effects are caused by changing parameter values, CircSim comes with a dialog to interactively change some parameter values during a running simulation. Figure 6.1 contains a screenshot of the interactive dialog.

The controls are grouped in several blocks. The block “Operation Control” on top contains all controls to operate a simulation. It can be started and stopped at arbitrary or predefined times. If it is stopped, the current simulation history and settings can be saved in a simulation state file. The button “Reset” removes the current simulation and enables to start a fresh simulation or to load a simulation state file for analyzing and/or proceeding. With the time slider the user can review the development of the simulation in the watch plot (see below) and generate plots of an earlier simulation time, although the simulation cannot be resumed from there. For the plot buttons see Section 6.3.2.

The middle control block on the left, “Initialization”, sets initialization parameters which are fixed during a simulation run. Those are the length of the road, number of lanes, vehicle density and population length. The latter limits the sec-

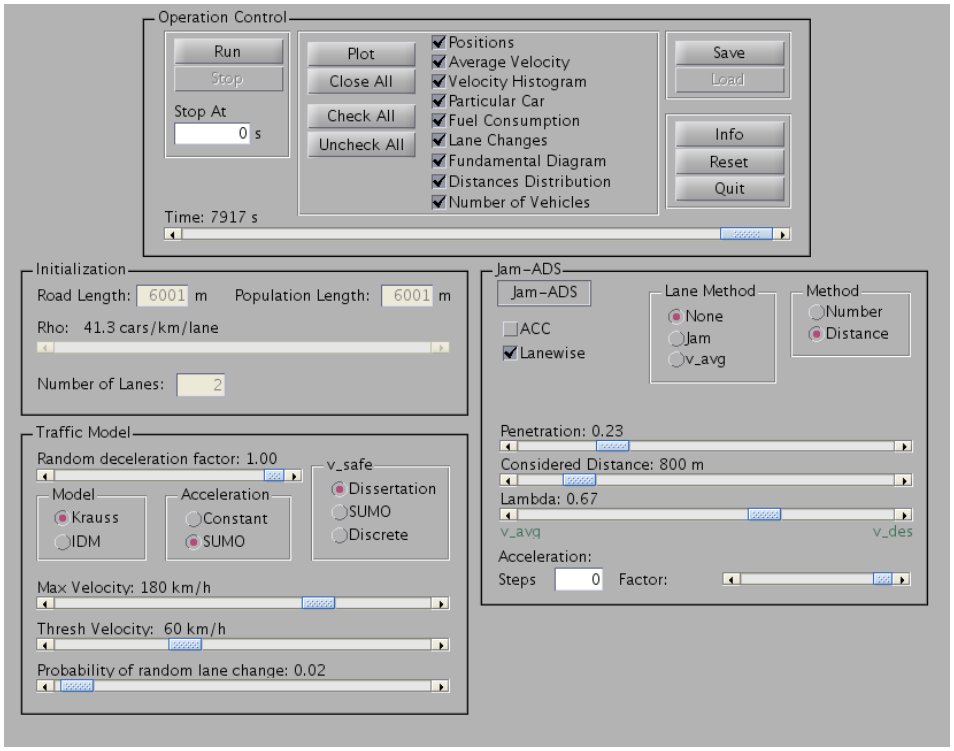


Figure 6.1.: Screenshot of the interactive dialog of CircSim

tion of the road which is populated initially. This allows to investigate the behavior of platoons with free road ahead. Changing the road length sets the population length to the same value to have a maximum population length by default.

The block on the lower left, "Traffic Model", controls the traffic model. All settings in this block can be changed during a simulation run. These are the meanings of the settings:

Model Current traffic model.

Random deceleration factor Random deceleration factor ε of the Krauß model (see Section 3.2.5).

Max velocity Maximum velocity of the road.

Thresh velocity Traffic jam threshold of the Krauß model.

Probability of random lane change Probability of random lane change in Krauß model (Section 3.3.1)

Acceleration Select between the constant maximum acceleration of the original Krauß model (Section 3.2.5) and the dynamic velocity-dependent maximum acceleration of SUMO (Section 6.2.1).

v_safe Select computation of v_{safe} in Krauß model. Choices are the v_{safe} of Krauß' dissertation (Section 3.2.5), v_{safe} as in SUMO (Section 6.2.1) and v_{safe} as suggested in Krauß, Wagner, and Gawron [89] ("Discrete") as explained above.

Settings which only belong to the Krauß model are invisible if IDM is selected.

The block on the right "Jam-ADS" controls parameters of Jam-ADS. The push-button "Jam-ADS" turns Jam-ADS on and off. The remaining settings are:

ACC Assume ACC equipment (Section 4.6), i.e., always include the predecessor into v_{avg} even if it is not equipped.

Lanewise Take v_{avg} only over the vehicle's current lane.

Lane Method Select a lane keeping strategy. Choices are no lane keeping, keep current lane if inside a jam, or keep current lane if v_{avg} of the preferred neighboring lane is approximately the same as on the vehicle's own lane. If one of the keep-lane methods is activated an additional slider appears to set its threshold.

Method Take v_{avg} over a specified number of equipped vehicles ahead or over equipped vehicles within a specified distance ahead. If number is selected the slider "Considered Distance" is replaced with a slider to set the exact number.

Penetration Change the equipment rate. This is done such that the equipment of the least possible number of vehicles changes. If it is increased, the appropriate number of randomly selected non-equipped vehicles becomes equipped. If it is decreased, randomly selected equipped vehicles lose equipment.

Considered Distance Distance ahead to take v_{avg} over. This control changes to number of vehicles ahead if number method is selected.

Lambda λ of Jam-ADS.

Acceleration Steps, Acceleration Factor Parameters to set consideration of acceleration of vehicles to average over. For explanation see Section 7.5.2 with an algorithm using these settings.

To see the effects of the current settings, CircSim provides a watch plot, which shows the vehicles as dots running in a circle. See Figure 6.2 for an example screenshot. The watch plot offers a menu to colorize the vehicle dots according to different properties like, for example, velocity, vehicle type, gap to the predecessor, Jam-ADS influence, or equipment. In addition, one can pick a vehicle in the watch plot to get its configuration and current state and to mark it in a different color to easily follow it. Furthermore, the picked vehicle's maximum velocity can be changed. This allows, for example, to slow down a vehicle to make it an obstacle.

The vertical bars on the right show the current average velocity and fuel consumption. The average velocity bar consists of a wide black bar in the background and two small bars in green and red in the foreground. The black bar displays the current average velocity of all vehicles. The green and red bars show the current average velocities of the equipped and non-equipped vehicles, respectively. Green for equipped and red for non-equipped vehicles is also used in several plots described below.

The average fuel consumption bars on the right are colored by the four parts of fuel consumption. As explained above, fuel consumption is split into an acceleration part (red), air-resistance part (blue), rolling-resistance part (green), and idle consumption part (black). Horizontally, the fuel consumption bar is split into the same bars as the average velocity bar. The outer small bars denote the average over all vehicles like the black average velocity bar. The inner left bar—currently sticking out on top—denotes the average over the equipped vehicles like the green average velocity bar and the inner right bar denotes the average over the non-equipped vehicles like the red average velocity bar

The small inner bars with average by equipment can be turned on and off for both velocity and consumption with a context menu inside the bars box.

Interactive changing of parameters is useful to develop some intuition, but it is not reproducible for systematic tests and parameter variations. To perform those, CircSim can also be controlled by scripts, which execute fully automatic simulation runs.

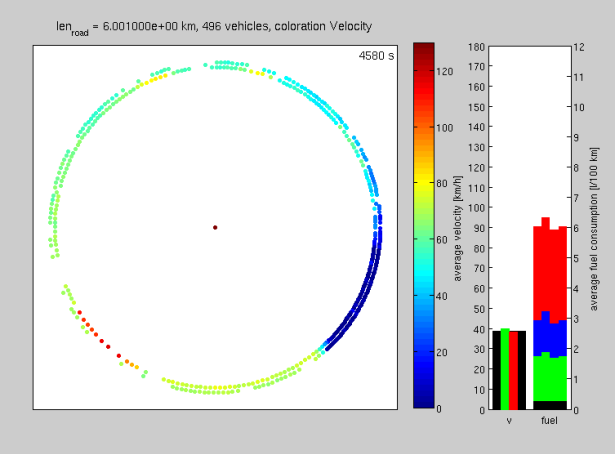


Figure 6.2.: Screenshot of CircSim's interactive watch plot showing a running simulation colored by current velocity. The color bar in the middle maps colors to velocities in km/h. In this case there is a traffic jam at the lower right of the circle and free traffic around the lower left of the circle. A few vehicles are going at nearly maximum velocity.

Further configuration settings

There are many more configuration settings not accessible in the control dialog. In fact, the control dialog is essentially a graphical editor for some of the configuration values. This section explains important remaining configuration parameters. More information on configuration settings is given in Appendix A.4.

The seed of the random number generator could be set to a fixed value or such that it is derived from system time to ensure different seeds for each run. A fixed value may be applied to avoid storing the complete simulation data and is sometimes useful for debugging, because with a fixed random seed simulations can be exactly reproduced.

An arbitrary number of vehicle types (also named classes) can be configured. Each type has its own name, maximum velocity, length, maximum acceleration, fuel consumption parameter settings, and equipment limit. The equipment limit is a factor of the general equipment rate and can be applied to define a vehicle type with less than the global equipment rate or no equipment at all. Finally, each vehicle type is assigned to either a fraction of the total number of vehicles or an absolute number of vehicles. The latter could be used, for example, to define single vehicles with extreme properties.

Reaction time and brake deceleration of the Krauß model can be set globally

for all vehicles. Decelerations are not configurable by vehicle type, because we assume the maximum deceleration is independent of the mass or other properties as it only depends on the friction of the tires on the road, which es nearly the same for all vehicle types. All parameters of the IDM can be configured as well.

A function to compute the maximum acceleration of the Krauß model has to be configured. CircSim provides a function for constant maximum acceleration always returning the maximum acceleration of the vehicle type in question and a function for maximum acceleration mimicking SUMO's dynamic maximum acceleration (Section 6.2.1).

To make the simulation more fine grained the length of a simulation step can be set. In principle, the simulation could be made coarse as well by increasing the length of a simulation step, but as mentioned in Section 3.2.5 the Krauß model is not collision free if simulation steps are longer than the reaction time and usually both are set to the same value. CircSim issues a warning if the simulation step length is configured to be longer than the reaction time.

During first tests with passenger cars and trucks on two lanes, we observed infinite passing maneuvers of trucks, because their speeds remained too close to each other to let any of them completely pass the other. To prevent this we introduced a small difference of maximum velocity on neighboring lanes. The leftmost lane has unchanged maximum velocity. For the lane right of the leftmost a velocity delta is subtracted from the maximum velocity, either the road's or the vehicle's maximum. This is repeated for every further right lane. Of course, the velocity delta can be set to zero to turn this feature off.

Traffic lights

As another feature we added traffic lights simulation capabilities to the Krauß model in CircSim. To model approaching a red traffic light, we added another dynamic maximum velocity limiting v_{des}, which depends on the current distance to the traffic light such that the vehicle stops with the configured normal brake deceleration in front of the light.

A traffic light can be in one of three phases: green, red, or turned-off. Green and turned-off lights are ignored, but displayed differently in the watch plot. The schedule is defined by three parameters, the start time and the lengths of the green and red phases. When simulation starts the light is turned off until its start time and then cycles infinitely through green and red phases beginning with green. The purpose of the turned-off state during the start time is to model phase shifts between different lights. Of course, the locations of the lights are also configurable.

The scheduling parameters can be changed during simulations either in the watch plot or by scripts driving automatic simulations. Even the start time can be set to a point in time after the current time to turn off a traffic light again for a while. The locations of the lights are fixed during a simulation.

We observed that the lane change model sometimes switches lanes of vehicles waiting at a traffic light. To prevent that, a minimum velocity for lane changes can be configured and is usually set to 5 km/h.

For realistic simulations of urban traffic, the maximum-velocity difference of neighboring lanes to prevent infinite passing maneuvers, described in the last section, should be turned off.

6.3.2. Types of plots

The watch plot already gives an impression how well traffic flows. However, this is not sufficient for a deep analysis. To support that, CircSim offers many different types of plots, which can be additionally fine-tuned.

The checkboxes in the upper middle box of the interactive dialog (Figure 6.1) allow to select which plots are opened when clicking on "Plot". "Check All" and "Uncheck All" are convenience buttons to quickly select or deselect all plots. "Close All" closes all currently open plot windows. Contrary, clicking "Reset" to start or load another simulation does not close plot windows, so this allows to compare plots of different simulation runs.

When control parameter values are changed during a running simulation, plots with a time axis are marked by straight lines at the times of parameter changes, drawn in a unique color. The legend explains which changes were done at the respective time.

We now explain all plot types in the order given in the interactive dialog.

Positions

The positions plot (Figure 6.3) presents the location on the road on the horizontal axis and the simulation time on the vertical axis, advancing from top to bottom. The locations on the left edge are neighbors of the locations on the right edge because of the circular road. For each vehicle at each simulation step a dot is drawn. A menu offers to select different colorizations of the dots: by equipment, by being in a jam, by vehicle type, by velocity, or by gap to the predecessor. The example screenshot is colored by velocity.

Furthermore, the plot can be limited to a single lane and to a certain simulation time range. Finally, an arbitrary vehicle selected by number could be highlighted, as can be seen in the example as well.

The example plot shows that soon after the start, a traffic jam developed with nearly standing vehicles while there was free traffic on the remaining road. It can be seen that the jam fronts moved upstream against the vehicles, like jams usually do. Even if the vehicle dots are not colored by velocity, jams always seem to look like cliffs on an otherwise flat land, because the gradients of the vehicle lines correspond to the momentary velocity.

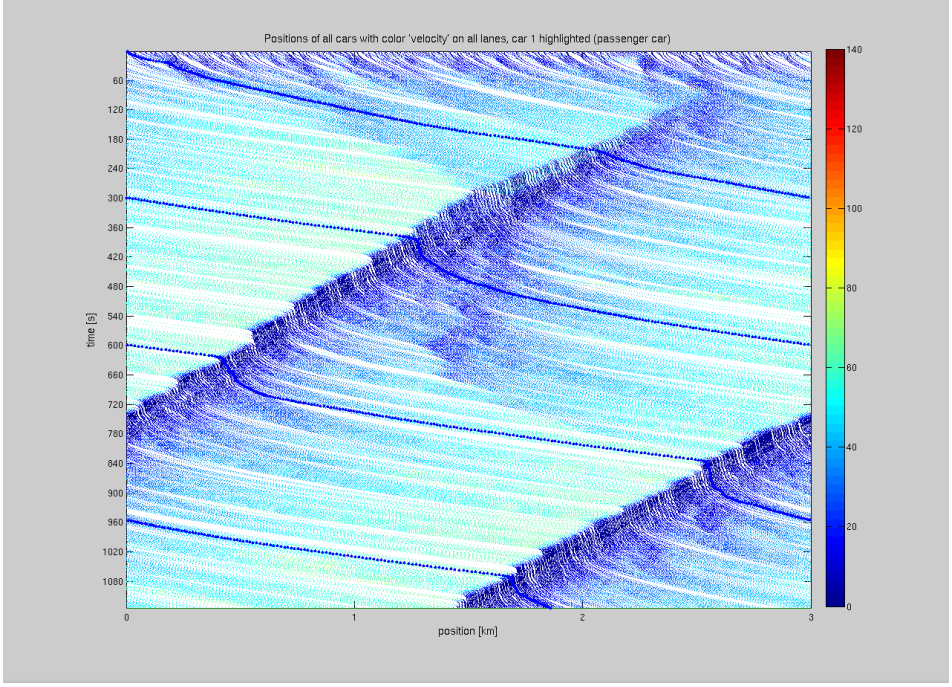


Figure 6.3.: Positions plot of CircSim colored by velocity. The color bar maps colors to velocities from zero at the bottom to 140 km/h at the top.

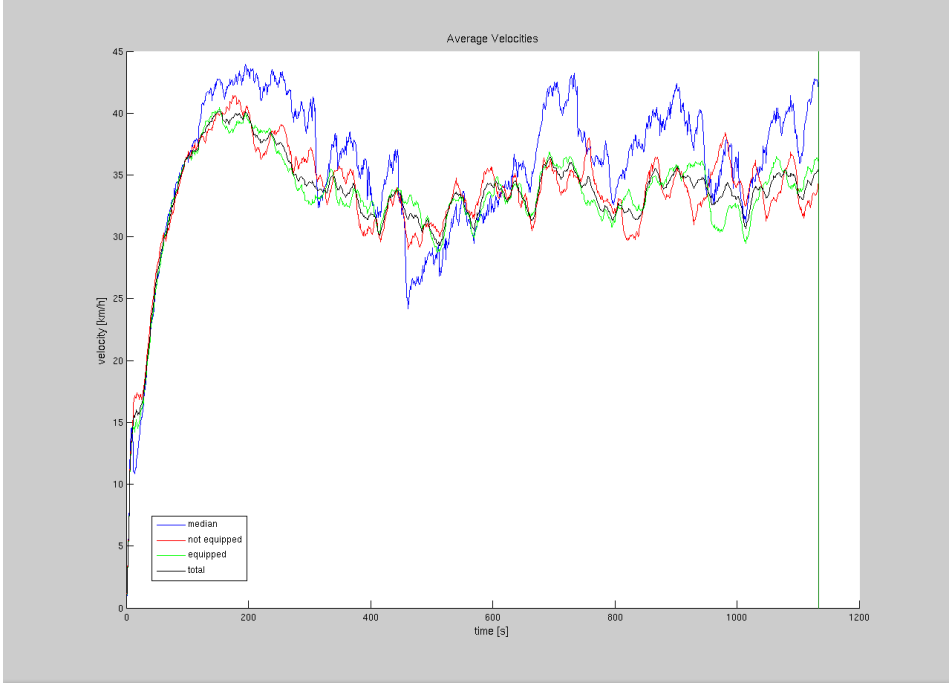


Figure 6.4.: Average velocity plot of CircSim

Average velocity

The average velocity plot (Figure 6.4) shows the time development of different velocity averages. The black line denotes the arithmetic velocity average of all vehicles at each simulation step. The green and red lines show the arithmetic velocity averages of equipped and non-equipped vehicles, respectively. In this example Jam-ADS was turned off and we have 60 % equipment rate, so the black line equals a sum of 60 % of the green line and 40 % of the red line.

The blue line exhibits the median velocity of all vehicles. The relation of the arithmetic average and the median gives an idea how evenly velocities are distributed over the vehicles. If the median is far below the arithmetic average, for example, the majority of vehicles is slow and only a few are very fast, which is the case if many vehicles are stuck in traffic jams while a few can move fast in the large remaining road sections between the jams.

The plot can be limited to average only over vehicles on a specific lane.

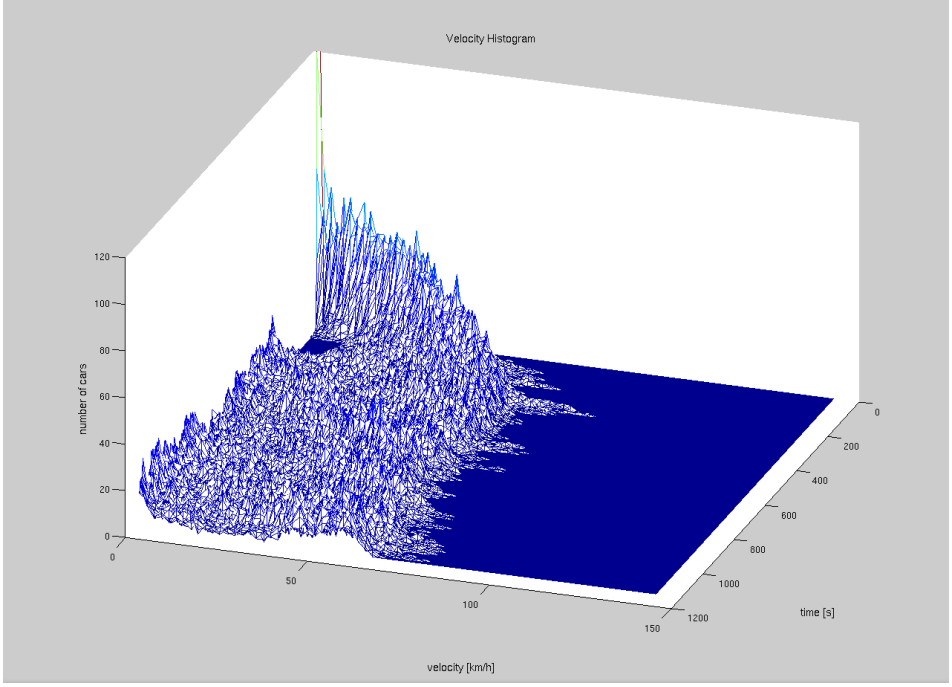


Figure 6.5.: Velocity-histogram plot of CircSim

Velocity histogram

The velocity-histogram plot (Figure 6.5) is a three dimensional plot showing the development of the velocity distribution over time. Time runs from back to front, velocity goes from left to right, and the z-axis shows the number of vehicles moving with a certain velocity at a certain time. Vehicle velocities are arbitrary double precision values, hence to get meaningful counts, velocities are ordered into bins of a configurable size, which was set to 2 km/h in the example screenshot.

The plot can be limited to count only vehicles on a specific lane.

Particular car

To see the time development of a particular vehicle, a plot is provided with four subplots each showing a vehicle property over time (Figure 6.6). The four properties are from top to bottom: velocity, fuel consumption, equipment, and lane.

The lines in the fuel consumption subplot have the same meaning for the particular vehicle as the respective lines of the fuel consumption plot, explained next.

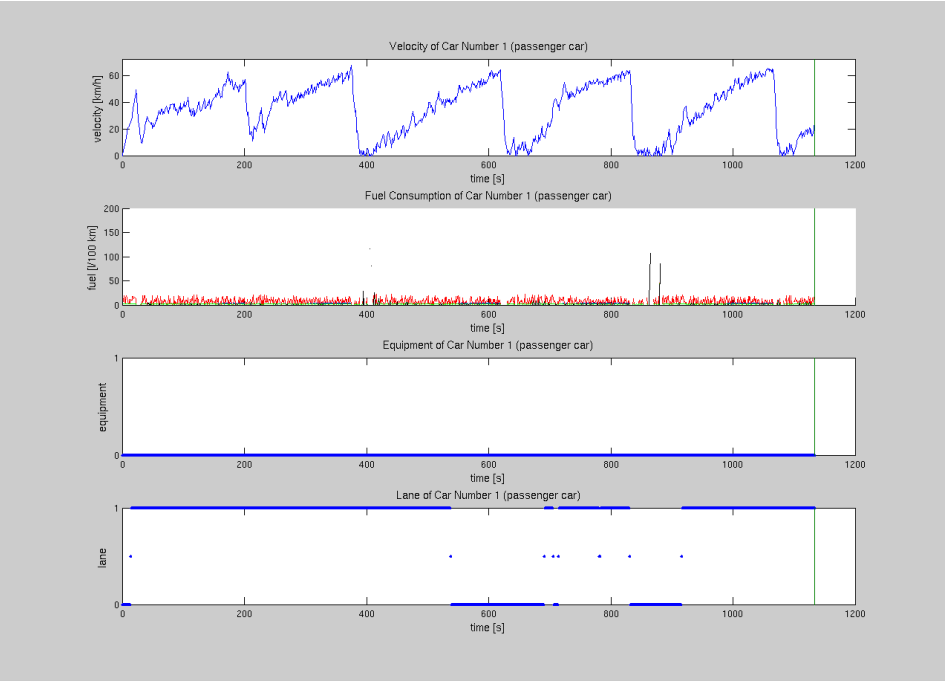


Figure 6.6.: Particular-car plot of CircSim

The equipment subplot could be interesting if the equipment rate was changed during the simulation run as explained in the description of CircSim's "Penetration" control in Section 6.3.1. The subplot shows if and when the particular vehicle was affected by that.

The lane subplot in the example shows a few data points on lane 0.5. As described in Section 6.3.1 vehicles changing the lane belong to two lanes for one simulation step. This is denoted by a non-integer lane number between departure and destination lane number. Lane numbers are counted from right to left starting at zero.

The particular-car plot is especially of interest in conjunction with picking a vehicle from the watch plot or the fundamental diagram plot, explained below. The number of the picked vehicle is announced and could be selected for the particular car plot to get more details about it.

Fuel consumption

There are two basic types of fuel-consumption plots, which can be selected by a menu option. Both show the average momentary fuel consumption of all vehicles over time. One fuel-consumption plot type also shows the averages for all equipped and for all non-equipped vehicles. The other fuel-consumption plot type draws the four consumption parts explained in Section 6.3.1 in a cumulative manner, i.e., each line represents the sum of the parts below (Figure 6.7). Furthermore, both fuel-consumption plot types can be limited to the vehicles of a specific lane.

Lane changes

This plot shows the number of lane changes over time (Figure 6.8). The black line is the total number of lane changes, the green line are the changes to the right neighboring lane and the red line the changes to the left neighboring lane.

The number of lane changes in each simulation step is usually a small integer changing at every step. Thus, the exact numbers oscillate too quickly to produce meaningful plots. To overcome this, the plot shows a moving average over time. A menu offers to change the number of simulation steps that the moving average spans. Its default can be configured and is usually set to 60, which corresponds to a moving average over the last minute, provided the simulation step length is set to one second as it is the case in all simulation runs presented in this thesis.

Fundamental diagram

The fundamental diagram in general is explained in Section 2.1, so this section only explains the special features of fundamental-diagram plots in CircSim.

Each vehicle at each simulation step has a local density computed by the space belonging to the vehicle and a local flow computed by its velocity over space.

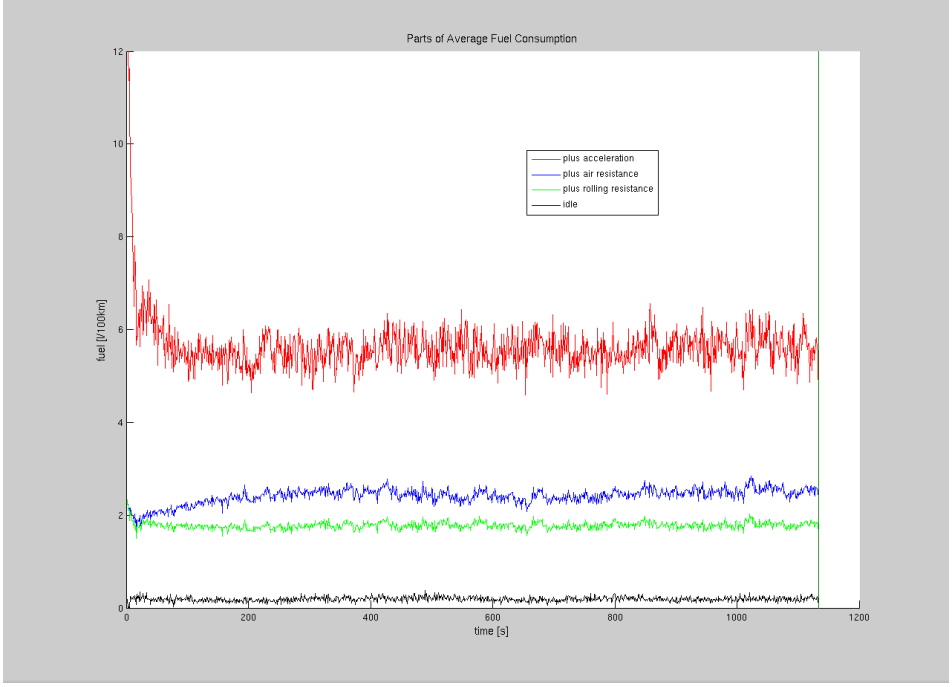


Figure 6.7.: Fuel-consumption plot of CircSim. The topmost line is the total consumption and the lower lines denote the borders between the consumption parts. From bottom to top they are: idle consumption (black), rolling-resistance consumption (green), air-resistance consumption (blue), and acceleration consumption (red).

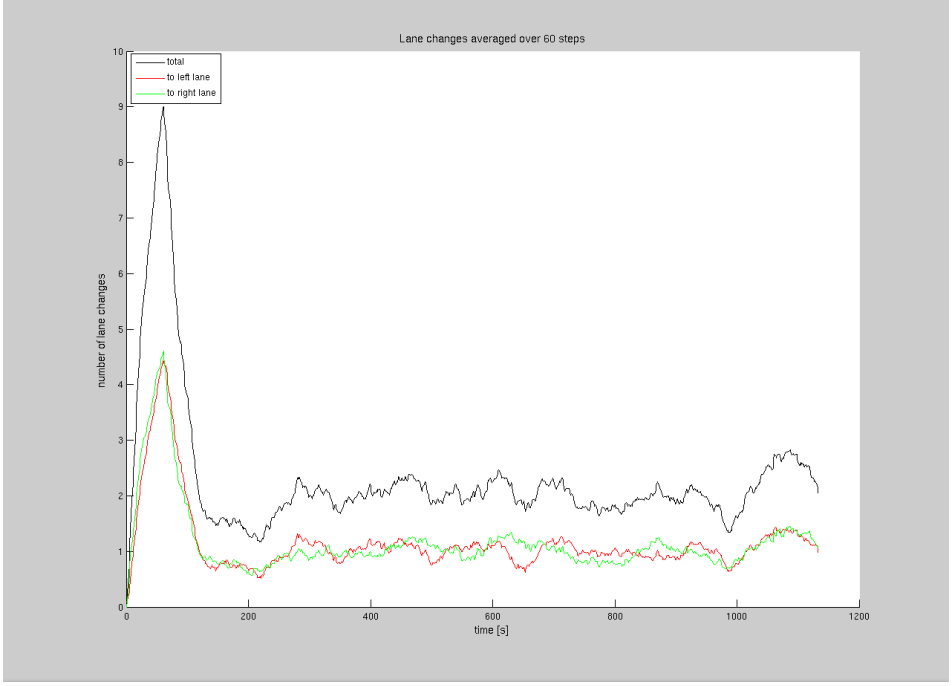


Figure 6.8.: Plot of lane changes in CircSim with moving average over 60 simulation steps.

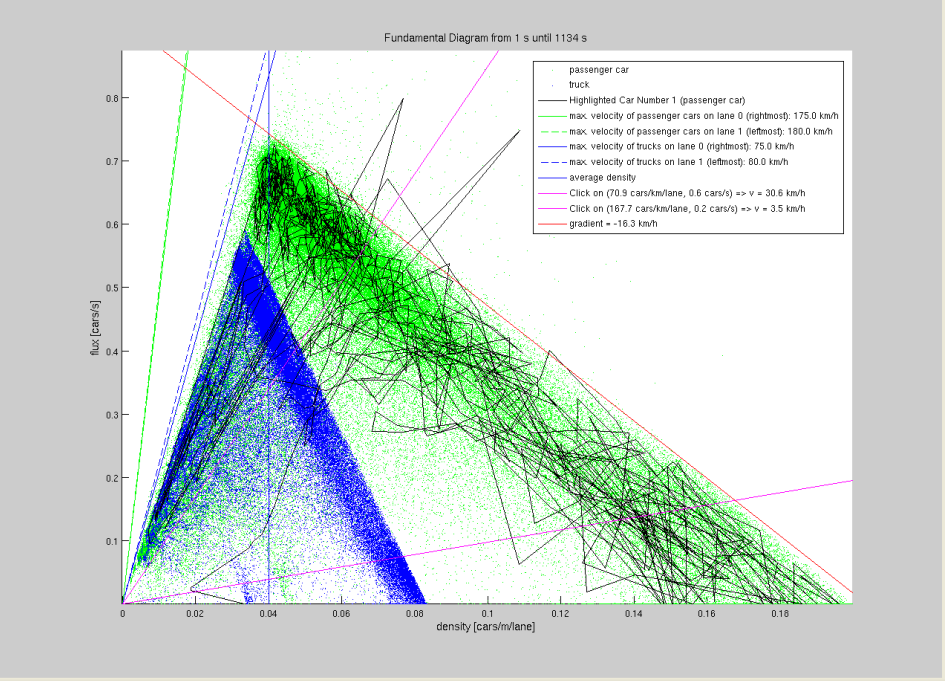


Figure 6.9.: Fundamental-diagram plot of CircSim

These local flow and density generate one data point in the fundamental-diagram plot for each simulation step (Figure 6.9).

In addition to the vehicles' data points the plot contains several straight lines. The vertical blue line at a certain density denotes the average density of all vehicles, which is fix for the whole simulation run. Half of the data points are on the left and half on the right of the average density line.[12]

In addition, there are several straight lines starting at the origin. These lines denote the maximum velocity of certain vehicles. The colors of the lines correspond to the vehicle types. As explained in Section 6.3.1 different lanes could be configured to have slightly different maximum velocities, which are distinguished by different line shapes. The legend explains to which vehicle type and lane each line belongs.

There are menu settings to assign different colors to the data points, either by vehicle type, by equipment, or by simulation time. The example is colored

[12] A pixel could be a pile of several data points, so this may not be obvious from looking at the plot.

by vehicle type with green for passenger cars and blue for trucks. Hence the corresponding maximum velocity lines are green and blue, as well.

Furthermore, it is possible to limit the drawn simulation time range. This allows to exclude the time until the system settled or to limit to the time between certain parameter changes to see their influence on the fundamental diagram. An example for this is Figure 7.4 in the next chapter that shows four different time slots of a single simulation run. Like many other plots, this can be limited to the vehicles of a specific lane as well.

As explained in Section 2.1, the gradient of a straight line in a fundamental diagram has the dimension of a velocity, which could of interest. To easily measure those gradients, the plot offers to pick two points and display the gradient between them. The screenshot gives an example with the straight red line above the jam branch. The purple origin lines intersect the red line at the picked points. The legend displays the coordinates of the picked points and the gradients of all three lines in km/h. It is also possible to pick only one point and get the gradient to the origin, which corresponds to the velocity of the picked data point. All such lines can be removed afterwards.

Finally, the user can pick a data point to highlight the trajectory of the vehicle the data point belongs to. This is also shown in the example screenshot. Alternatively, the user can enter the number of a particular car to highlight its trajectory. The number could be obtained by picking a vehicle in the watch plot. Of course, the vehicle highlighting can be removed as well.

The fundamental diagrams of CircSim exhibit two structures which are not seen in empirical fundamental diagrams. One is the existence of several clearly separated jam branches and the other is the sharp upper border of those jam branches.

The two separate jam branches in the example are caused by the artificial fact of having exactly two vehicle types in the example configuration, both with fixed property values of which some are considerably different. The colors of the data points in the example plot show that the upper jam branch belongs to passenger cars and the lower to trucks. Adding a third different vehicle type produces a third distinguishable jam branch. Contrary, in reality we always have a spectrum of vehicle properties because of the many different vehicle types and individual driving behaviors. Thus, we could produce more realistic fundamental diagrams by configuring many different vehicle types, but for our purposes it is sufficient to know the source of the different jam branches.

The other artificial structure, the sharp upper border of the jam branch, is not absolute. There are usually a few data points above. Section 3.2.5 derives the gradient of the stationary jam branch in Equation (3.15). The sharp upper borders correspond to the stationary jam branches of the configured passenger cars and trucks depending on the general reaction time and the different vehicle

lengths.[13] Again, in reality we have a distribution of temporal gaps and vehicle lengths, which blurs the borders of the jam branches.

As explained in Section 2.1 the gradient of the jam branch reflects the usual upstream speed of a jam front. The distinct jam branch of trucks with a gradient around −40 km/h in our example gives room for the speculation that a jam front in pure truck traffic would travel upstream that fast. However, we do not know of any empirical data to support that guess.

Distance distribution

The distance-distribution plot (Figure 6.10) shows the distribution of either temporal or spatial headways. The spatial headway is the gap to the predecessor and the temporal headway is the time it would take to pass the spatial headway at the current velocity. Beside the distribution over all vehicles, the plot can display different distributions for either each lane, each vehicle type, or equipped and non-equipped vehicles. It can also be limited to a given time range or a specific lane. Of course, the latter makes no sense when showing lane-wise distributions, but there is no reason to disable that in CircSim. The granularity of the distribution can be set as well.

Numbers of vehicles

The last plot displays the number of vehicles on each lane over time (Figure 6.11). Beside the total number of vehicles the dotted lines display the numbers of equipped vehicles on the respective lanes. The number of unequipped vehicles is not shown to prevent cluttering the plot with too many lines. It can be seen as difference between the dotted and the corresponding solid lines.

6.3.3. Validation

Every newly developed (or fundamentally changed) software needs validation to prove its reliability. On the other hand, simulation software cannot be tested in a strict sense or formally proven, because there is no pre-known relation between input and output, otherwise the simulation would be obsolete. This applies particularly if the simulation contains randomness to compensate the lack of knowledge or to simplify the simulation model.

However, we can compare statistically aggregates like averages or value distributions to known data of real traffic or other simulators like it is done by Lownes and Machemehl [101] for the implementation of the Wiedemann model in VISSIM. Comparison of results of different simulators helps to distinguish between implementation bugs and deficiencies of the simulation model itself.

[13] This is also a sign of a correct implementation of the Krauß model in CircSim.

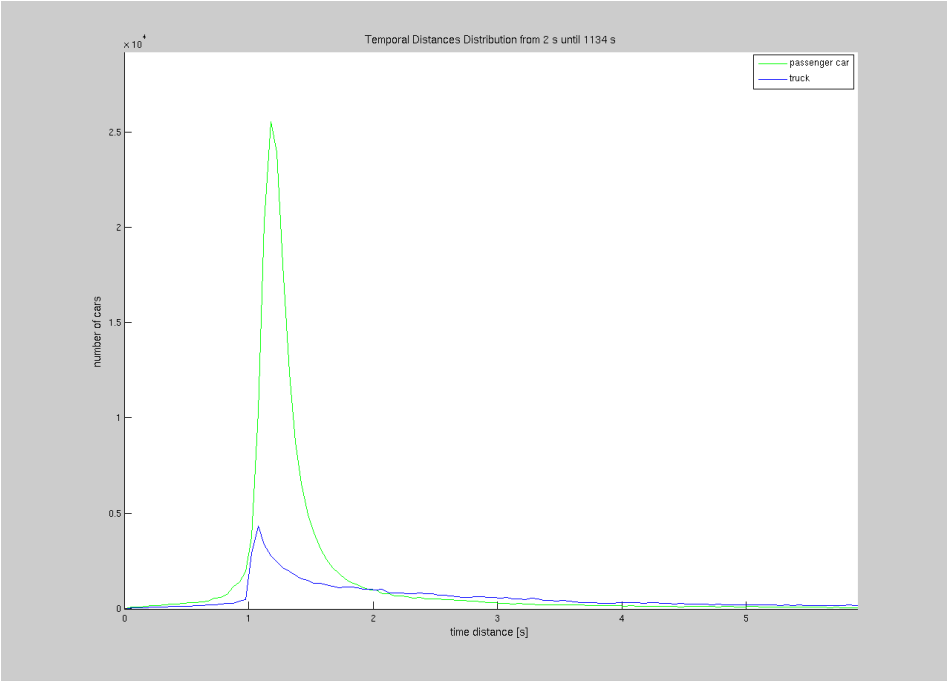


Figure 6.10.: Distance-distribution plot of CircSim with temporal headways

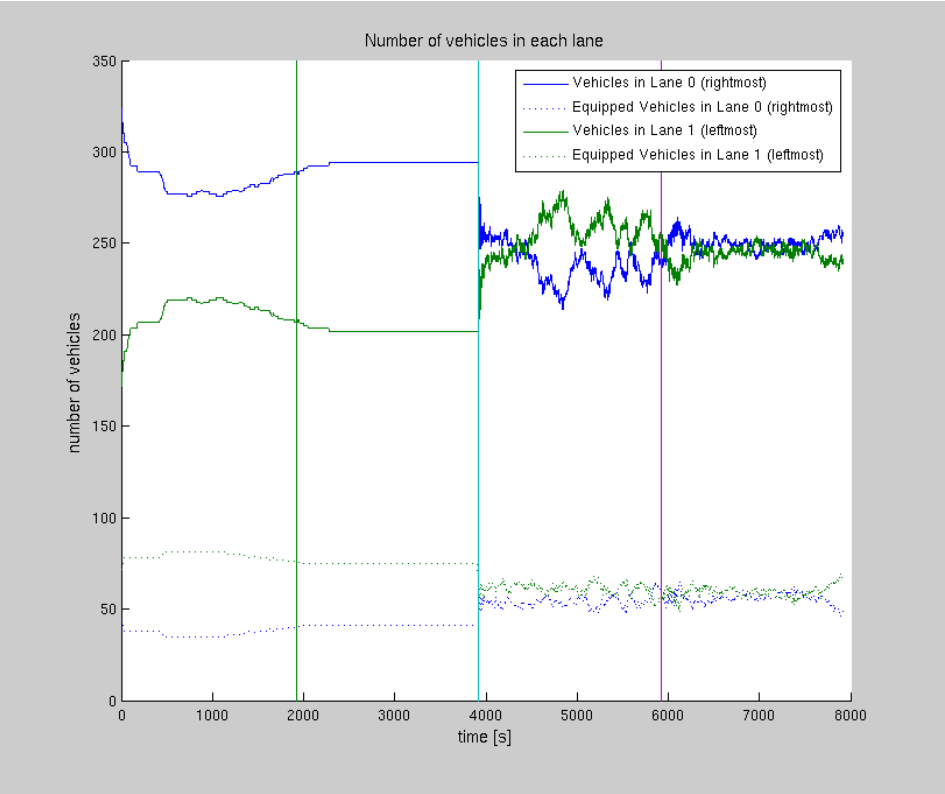


Figure 6.11.: Plot of number of vehicles per lane of CircSim

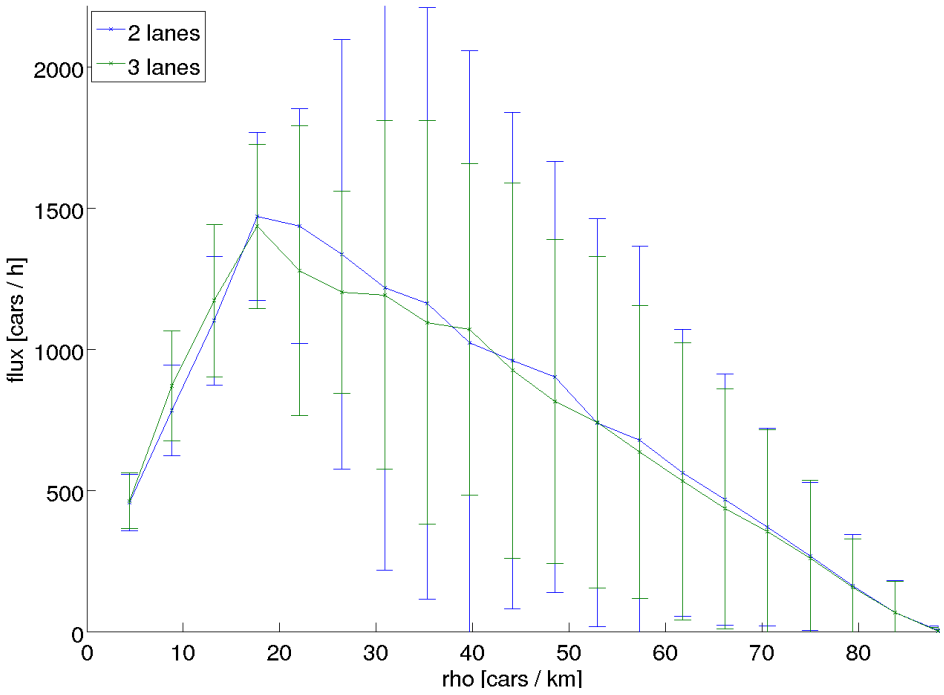


Figure 6.12.: Fundamental diagram with one simulation run per data point. The density (rho) is the average density of vehicles on the circular road and the vertical bars denote the range of observed flows. The connecting lines denote the average flows.

An important statistically aggregated result is the gradient of the jam branches in fundamental diagrams, that corresponds to the velocity with which jams travel upstream as explained in Section 3.1. According to Kerner and Rehborn [78] and Kerner [68] this velocity ranges between $-15\,\mathrm{km/h}$ and $-19\,\mathrm{km/h}$.

All fundamental diagrams generated with CircSim have a jam branch with a gradient in this range. This does not only apply to fundamental diagrams of a single run (Figure 6.9), but also to fundamental diagrams of a simulation session with one simulation run per data point, like Figure 6.12.

More evidence comes from an experiment with 22 real vehicles moving in one lane in a circle with 230 m circumference published by Sugiyama et al. [132]. Their observations are in accordance with the simulation results of CircSim.

Another validation approach is the comparison of the results of the Krauß model versus the IDM in CircSim. Both produce similar results. Additionally, we executed many simulations of the closed system with both SUMO/Shawn and

CircSim with the same configuration and comparable results. See Chapter 7 and particularly Section 7.6 for examples.

6.4. Control and evaluation scripts

For easy and reproducible control of simulations we developed several control scripts in Perl[14] and Python[15]. In addition, we developed several scripts for evaluation of the results by producing plots and ease browsing through a huge amount of plots. We describe some of their features here to explain in more detail how the simulation results of the following chapters are produced. The scripts were continuously improved during this work, thus not all features were available during the early stages of this work. However, the presented results stem from the more recent simulation sessions.

After first experiments with the interactive features of CircSim we started to run complete simulation sessions controlled by a script to vary parameters of interest. A simulation session is usually executed by one control script call, which automatically executes all simulation runs with different values of the model-parameter set. For some simulation sessions the control script is called several times to repeat crashed simulation runs after a bug has been fixed or to extend the set of included model-parameter values.

There are two generations of simulations sessions corresponding to the closed and the open system simulations presented in the following chapters. In addition, we developed a script to visualize some analytical relations, which we applied to render some of the plots in Chapter 9.

6.4.1. Closed system scripts

The scripts described in this section produced the results presented in Chapter 7. They are implemented in Perl, Python, and MATLAB. Figure 6.13 gives an overview of the entire script chain and the involved data files.

Each simulation run starts with SUMO/Shawn and proceeds with CircSim. For both simulators the run begins without Jam-ADS and turns it on later. Thus, a simulation run consists of four different time slots.

A configuration file `run.cfg` contains the model-parameter values for the different simulation runs of a sessions and some general data like, for example, the lengths of the time slots. The script `run-shawn-sumo-circsim.py` reads this configuration file and executes all simulation runs of the session and finally takes care to produce a summary of the simulation runs for comprehensive evaluation of the

[14] `http://www.perl.org/`
[15] `http://www.python.org/`

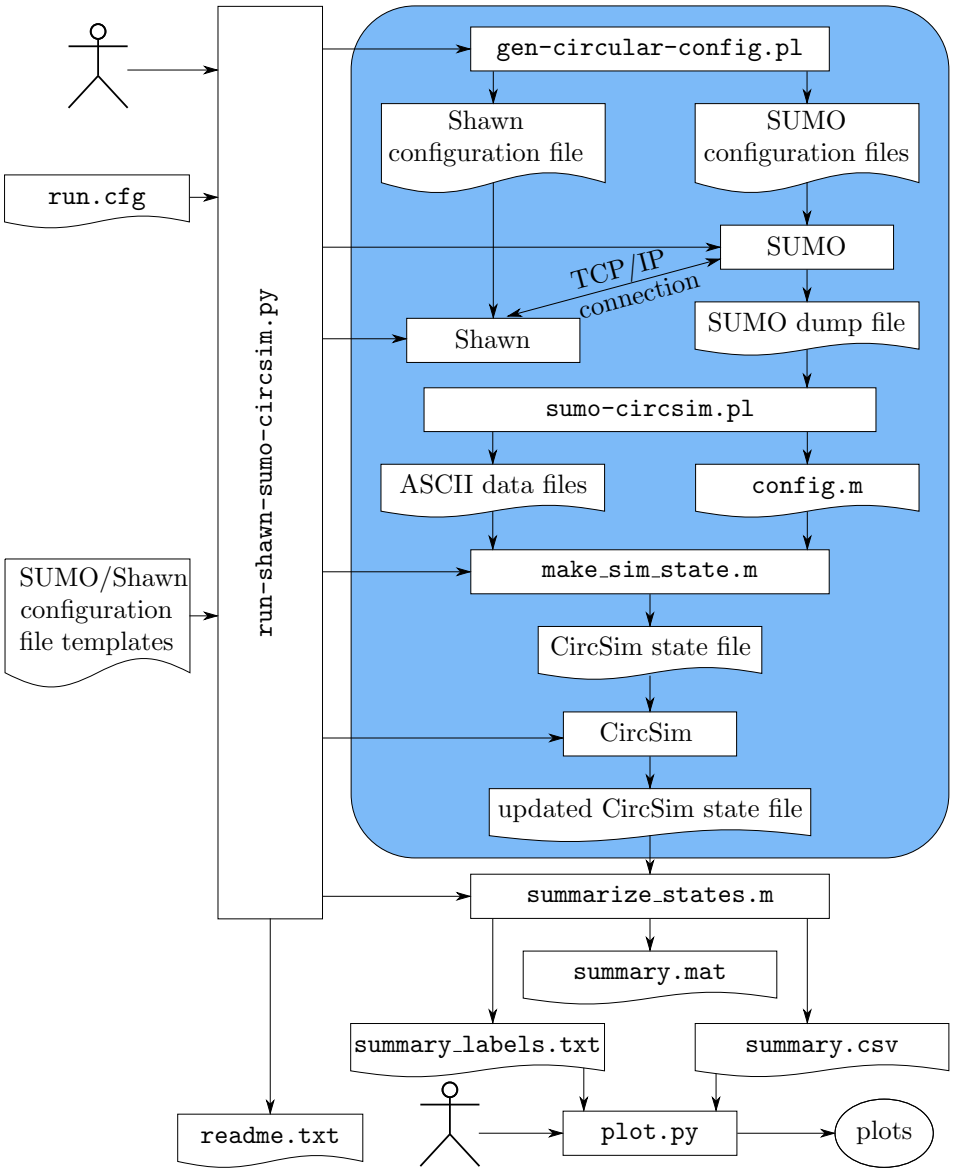


Figure 6.13.: Script chain to execute and evaluate simulation sessions for the closed system. The part with the blue background is executed once for each single simulation run, while the parts outside treat a complete simulation session. The stick figures indicate manual starting of the scripts pointed to.

entire session. To do all this `run-shawn-sumo-circsim.py` calls sub-scripts for particular tasks.

For each simulation run `run-shawn-sumo-circsim.py` makes copies of templates of SUMO's and Shawn's configuration files. These copies of the configuration templates are patched with the special values of the actual simulation run. In case of SUMO this is done by the sub-script `gen-circular-config.pl` generating the circular road for SUMO, which consists of four junctions and edges, such that it has the configured total length and number of lanes. In addition, `gen-circular-config.pl` generates vehicle types, vehicles, and routes for SUMO.

Next, `run-shawn-sumo-circsim.py` starts SUMO as TraCI server and Shawn as TraCI client such that both communicate via TCP/IP over a local loopback interface as described in Section 6.2.3. While performing the traffic simulation, SUMO puts all micro-states into a so-called dump file.

When SUMO and Shawn terminate, `run-shawn-sumo-circsim.py` calls `sumo-circsim.pl` to convert the micro-states in SUMO's dump file into several CSV[16] files and a CircSim configuration file. Those can be read by the MATLAB script `make_sim_state.m` and converted to a CircSim simulation-state file (Section 6.3.1). While doing this, `make_sim_state.m` resets the last micro-state to a uniform state of all vehicles as explained in Section 7.2. Lastly, `run-shawn-sumo-circsim.py` calls CircSim to resume the simulation run.

SUMO, Shawn, and CircSim are single-threaded applications, thus they employ only one processor core at a time. This even applies to the combination of SUMO and Shawn, because they alternately execute their simulation parts. To make use of all processor cores for a simulation session `run-shawn-sumo-circsim.py` is able to perform several simulation runs in parallel threads. The number of parallel threads has to be set in `run.cfg`. To make `run-shawn-sumo-circsim.py` multi-treaded we developed a generic module `workthreads.py` to manage a dynamic number of work threads, which execute arbitrary Python functions.

Finally, when all simulation runs of the session are finished `run-shawn-sumo-circsim.py` calls the MATLAB script `summarize_states.m`. It aggregates each single simulation run of the session such that it computes average velocities and fuel consumptions for each of the four time slots: SUMO/Shawn without Jam-ADS, SUMO/Shawn with Jam-ADS, CircSim without Jam-ADS, and CircSim with Jam-ADS. The results are stored in the CSV file `summary.csv` for further processing or conversion to a spreadsheet. Additionally, a file `summary_labels.txt` is generated with labels for the columns of `summary.csv` and the MATLAB data file `summary.mat` containing the complete summary data for further processing with MATLAB. Note, that averaging the fuel consumption over a time range has same peculiarities explained in Section 6.4.4.

[16] CSV stands for "comma separated values", a common ASCII-based file format for tabular data.

While doing all this, `run-shawn-sumo-circsim.py` writes a general simulation-session description into `readme.txt`, which contains several timestamps, the content of `run.cfg`, and some success statistics.

The user calls `plot.py` to render the data of `summary.csv` with the help of `summary_labels.txt` into the plots explained and presented in Section 7.6 to show how the aggregated data of the simulation-runs depend on certain model-parameters.

The on-line version of the plots generated with `plot.py` prints all model-parameter values and the belonging simulation runs of each displayed line when clicking on it. All lines belonging to the same simulation run but different time slots are highlighted by this as well.

6.4.2. Open system scripts

The scripts of this section executed the simulation sessions of the open system (Chapter 8). The open-system simulations are done solely with SUMO/Shawn, because the open system has a dynamically changing number of vehicles on the road, but CircSim's data structures are based on a fixed number of vehicles (see Appendix A.1), thus extending CircSim to simulate an open system would have required a complete rewrite.

All open system scripts are written in Python. For plotting the library MatPlotLib[17] is applied, which is based on the library Numpy[18]. Python together with Numpy and MatPlotLib offers features and performance similar to MATLAB and with similar data structures. Opposed to MATLAB, Python, Numpy, and MatPlotLib are distributed under open-source licenses similar to the GPL [43], hence they are freely available.[19]

The script chain of the open system is less complicated than the script chain of the closed system, because there is no need to convert data structures between different simulators. The main script to perform an open-system simulation session is `mastercontrol.py`. Like `run-shawn-sumo-circsim.py` it reads a configuration file `run.cfg` and additionally a configuration file `general.cfg`, that contains configuration settings shared with the plot script `plot.py`, described below. `mastercontrol.py` reads, copies and patches templates of SUMO and Shawn configuration files, like the corresponding closed-system script `run-shawn-sumo-circsim.py`. Furthermore, `mastercontrol.py` applies also `workthreads.py` to execute SUMO/Shawn simulations in several parallel threads.

`mastercontrol.py` applies the helper modules `basic.py` and `linear.py`. The latter contains code depending on the particular road-network topology. It could

[17] `http://matplotlib.org/`

[18] `http://numpy.scipy.org/`

[19] There are plans to port CircSim to Python with MatPlotLib and Qt as GUI toolkit to become independent of a commercial license.

be replaced to simulate a closed system or any other topology. `basic.py` contains code shared with the plot script `plot.py` described below. A main part of the topology-dependent module `linear.py` is the conversion between SUMO positions, which consist of an edge ID and an offset, to an absolute position. The absolute position is needed for plotting and to define the borders of HDCs in the open system (Section 8.3).

The settings of model-parameters is more flexible in `mastercontrol.py`. Each section of model-parameter settings in `run.cfg` may explicitly include further such sections, which could be applied to make `mastercontrol.py` loop over the includes to gain the model-parameter values for each single simulation run. This feature allows to configure arbitrary sets of model-parameter values by inserting sufficiently many intermediate model-parameter value sections to avoid any redundant repetition of parameter-values. Contrary, `run-shawn-sumo-circsim.py` always loops over all possible combinations of model-parameter values.

The primary results of `mastercontrol.py` are two files for each simulation run. One contains all meta-data of the simulation run and the other the complete history of micro-states. Both are in Python.cPickle[20] format that allows quick processing by further Python code. In addition, in case of the micro-states it needs much less disk space than the original netstate-dump files of SUMO, because those are a highly redundant XML files.[21] Beside those, `mastercontrol.py` writes a simulation-session description file `readme.txt` with general information about the simulation session like configuration settings, software versions, timestamps, and success statistics and it writes a file `errorlog.txt` if an error occurred, to ease debugging.

To evaluate the results of a simulation session, we developed another script, `plot.py`. It has its own configuration file `plot.cfg` beside the general configuration file `general.cfg` shared with `mastercontrol.py`. `plot.cfg` contains the types of plots to produce, plot limits, fuel consumption parameters, and the location of the *evaluation section* (Section 8.1), whose aggregated data some plots exhibit. `plot.py` shares the code of the modules `basic.py` and `linear.py` with `mastercontrol.py`, which can be configured to call `plot.py` when the simulation session itself has finished.

`plot.py` starts several plotting processes similar to the simulation threads of `run-shawn-sumo-circsim.py` and `mastercontrol.py`. We need to apply independent OS-processes for plotting, because MatPlotLib is not multi-threading-safe. This makes output handling and finish time estimations for e-mail notifications more complex.

[20] Python.cPickle is a standard library of Python to serialize Python data structures into a binary file format.

[21] Our intermediate netstate-dump files were up to 1 GiB large and the resulting Python.cPickle files only around 600 MiB. With hundreds of single simulation runs this is an important saving of disk space.

To produce the plots of a simulation run, `plot.py` reads the two files of the simulation run produced by `mastercontrol.py`, as explained above, and renders—depending on the configuration—up to the following types of plots:

Positions Like the positions plot of CircSim colored by velocity. There is one plot for each lane and one for all lanes together.

Velocity Delta Like the positions plot, but colored by velocity changes at each simulation step.

Jam-ADS Delta Like the positions plot, but colored by velocity changes induced by Jam-ADS.

Average Velocity Time development of arithmetic average and median velocity within the evaluated section.

Fuel Consumption Like the average fuel-consumption plots of CircSim. Like in CircSim there is one plot with consumption, split into its four parts and one plot distinguishing equipped and non-equipped vehicles. Opposed to CircSim, the fuel consumption is smoothed by a moving average. A moving average shows at each time step an average over a fixed number of previous time steps.

Edge Flows Time development of inflow and outflow of different road sections. Because of the discrete number of vehicles and simulation steps there are large short-term fluctuations, thus the plot exhibits a moving average as well.

Lane Flows The same as edge flows but per lane instead of per edge.

Density Density of equipped, non-equipped, and all vehicles within the evaluated section over time. Like the positions plots there is one plot for each lane and one for all lanes together.

Travel Times Average time it takes the vehicles of each vehicle type to pass the evaluated section with a moving average. For each simulation step the average travel time is taken over the vehicles that just passed the evaluated section. Because of integer vehicle numbers this value has large fluctuations like the flows, thus the moving average is plotted. It is possible to combine vehicle types to groups such that each group is treated like a single vehicle type. Groups are defined in the plot configuration by Perl regular expressions.[22]

[22] *Perl regular expression* is a common name for the sophisticated regular expression language introduced by the language Perl, meanwhile adopted by many other programming languages.

Total Fuel Consumption Mean fuel-consumption parts of the vehicles within the evaluated section with a moving average. This is the same as the travel times plot except that the mean consumption within the evaluated section is measured.

Moving averages are usually taken over the last 60 seconds. This can be configured in `plots.cfg`. Fuel consumptions are computed according to Section 6.4.4.

Some of these plots are generated for each lane beside a plot covering all lanes. Hence, a simulation session of the open system generates a huge number of plots, up to several thousands. To make evaluation of the plots of a simulation session easier, we developed a plot browser, that allows to select plots by type and model-parameter values and display the selected plots side-by-side.

Rendering the positions plots of the open system with MatPlotLib has a special problem. MatPlotLib as a vector graphics tool stores a special data structure for each single data point and renders it as a small circle. The huge number of data points took too much memory and the small circles with their anti-aliasing made the plot nearly unusable. To solve this, `plots.py` renders the data points directly into a bitmap and adapts the plot's axes to it.

6.4.3. Analytical plot scripts

To support the analytical imagination of the effects of Jam-ADS we would have liked to have diagrams with more than three dimensions. To address this, we developed a Python script, that applies the GUI toolkit Qt[23] and MatPlotLib to generate two- or three-dimensional diagrams with additional interactive parameters. Those are realized as slider controls, checkboxes or list selections, which directly change the diagram. The sliders act like moving through additional dimensions. This tool is applied to generate the diagrams presented in Chapter 9.

6.4.4. Fuel consumption

The details of fuel consumption computation presented in this section apply to the closed and the open system, thus we explain them together here.

Equation (6.8) expresses the fuel average of CircSim. It computes the average fuel consumptions of all vehicles at a single simulation step. However, the interesting average consumption of a single vehicle over a longer time range is the total consumption over the total distance, which is in general different to the arithmetic average of the momentary consumptions of a vehicle. A little example proves this: Imagine a vehicle moves half of its travel time at 30 km/h consuming 12 l/100 km and half of its travel time at 130 km/h consuming 6 l/100 km. Then the

[23] `http://qt.digia.com`

mean consumption is 7.125 l/100 km while the arithmetic average of momentary consumptions is 9 l/100 km.

The meaningful average consumption over several vehicles is nevertheless the arithmetic average. Thus, we compute the regular fuel consumption part $p \in \{\text{idle}, \text{roll}, \text{air}, \text{acc}\}$, of a set of vehicles over a time range from t_1 to t_2 as

$$C_p(t_1 \to t_2) = \frac{1}{|\{\text{vehicles}\}|} \sum_{v \in \{\text{vehicles}\}} \frac{\sum_{t'=t_1}^{t_2} c_p(v, t')}{\sum_{t'=t_1}^{t_2} d(v, t')} \tag{6.9}$$

with $c_p(v, t')$ as consumption part p of vehicle v during time step t' and $d(v, t')$ as distance traveled by vehicle v during time step t'.

This expression is not entirely applicable if a vehicle does not move during the observed time range, because it makes the denominator of its summand vanishing. For the summary data of the closed system generated with `summarize_states.m` (Section 6.4.1) we assume that during the aggregated long time ranges this is negligible, thus we only consider vehicles, that actually move during the aggregated time range. In case of the total fuel consumption plot of the open system (Section 6.4.2) we know that the vehicles have moved, because we only consider vehicles that passed the evaluation section, thus the problem does not occur in that case.

In case of the regular fuel-consumption plot of the open system (Section 6.4.2) we cannot neglect non-moving vehicles, because the short time range of the moving average may include many non-moving vehicles, particularly when traffic jams are included. Furthermore, we want to compare fuel consumptions of different simulation runs with and without traffic jams, thus ignoring the consumption of stopped vehicles would tamper the results. To solve this, we sum up the absolute consumptions as long as a vehicle stops and add it to the absolute consumption of the first simulation step at which it moves again. Due to the moving average this does not produce too large discontinuities.

For the summary data of the closed system we compared the results of Equation (6.9) with the arithmetic average of the momentary fuel consumptions of some simulation sessions. The difference exceeded rarely one percent and was usually much less.

6.4.5. Generate road networks and other configurations

The configuration settings of CircSim are designed to be as detailed as necessary for this work, but SUMO with its arbitrary road network and dynamic number of vehicles has many more configuration settings. In fact, SUMO comes with several preparation tools to convert high-level configuration files to low-level, partially redundant configuration data files, which SUMO needs.

One of these tools is `netconvert`, which expects at least one XML-file with a list of road junctions and another XML-file with a list of edges connecting the junctions. In addition, the XML-attributes of edges can be combined to edge types, that are passed to `netconvert` in a third XML-file. Special lane connections can be specified in a fourth XML-file. All these files have to be properly set up for each road network, so we needed to automate this to incorporate different road lengths and numbers of lanes in simulation sessions.

For both of our simulation scenarios (closed system and open system) we prepared generic input files for `netconvert` with the basic network structure and we developed scripts to adapt those generic network files to the global network settings of a specific simulation run. For example, in case of a circular road the generic network consists of four junctions and edges, which our script places such that their total length equals the required circumference of the circle.

Next, SUMO needs so-called route files to configure the vehicles. A route file contains one data set for each single vehicle to simulate, except that a fixed number of clones of a given vehicle could be automatically generated periodically. The automatic repetition of vehicles is sufficient for our closed system scenario with a fixed number of vehicles. For the open system scenario we apply SUMO's tool `duarouter` to convert given inflows of each vehicle type to a route file suitable for SUMO itself.

We also developed scripts to patch generic route input files to reproduce the vehicle demand of a specific simulation run of the open system.

6.4.6. Versioning

All tools, simulators and scripts, are put under version control using Git[24]. We extended SUMO and Shawn by a command line option to make them return the Git reference (SHA1) of the exact revision they were compiled with and an indicator if some source files were different to the referenced Git commit.

This is applied to insert the Git references of SUMO and Shawn beside the Git references of CircSim (in case of the closed system) and the Git reference of the respective script itself into the simulation session description file `readme.txt` written by `run-shawn-sumo-circsim.py` for the closed system (Section 6.4.1) or by `mastercontrol.py` for the open system (Section 6.4.2).

Provided a simulation session is configured to apply fixed random seeds, the session configuration and the Git references in `readme.txt` are sufficient to reproduce the entire simulation session with the exact same micro-states. Thus, it is not crucial to back up all the data of a simulation sessions as long as `readme.txt` is safe.

[24] `http://git-scm.com`

In addition, CircSim's simulation state files (Section 6.3.1) contain the Git reference of the CircSim version they are generated with. This allows to select the appropriate version of CircSim to open a simulation state file, even if the format has changed by further development of CircSim.

7. The Closed System

To find out if and how well Jam-ADS works, we start with a numerical simulation of a circular road of fixed length. This allows us to control the vehicle density and avoid artificial effects caused by randomly adding and removing vehicles.

Beside many interactive and automatic pretests we performed 21 documented simulation sessions of the closed system, which took more than 1660 hours of computation and whose data occupies 239 GiB of storage space. Many of those sessions were executed with SUMO/Shawn and CircSim in each simulation run as described below in Section 7.2. In the following, we present a few selected examples from these simulations.

7.1. Interaction of random deceleration and Jam-ADS

An early observation made with pure CircSim simulations is the effect of changing the order of random deceleration and Jam-ADS. Jam-ADS is much more efficient if it is performed after the random deceleration as we did initially because the random deceleration belongs to the Krauß model and Jam-ADS is an additional strategy we want the driver to follow. However, Jam-ADS reduced the velocity to a predetermined value, which annihilated most of the randomness introduced by random deceleration, but this randomness is the main cause of traffic jams. Thus, Jam-ADS after random deceleration reduced efficiently but unrealistically the amount of traffic jam.

To resolve this and be more realistic, we exchanged random deceleration and Jam-ADS, because we assume drivers produce the same amount of randomness when following Jam-ADS as when following their own desired velocity.

7.2. General simulation configuration

This section explains the general settings of the simulations presented in this chapter. Specific settings for particular simulations are described together with the results.

We developed CircSim to simulate the closed system, hence the circular road is hard-coded into CircSim. In SUMO we built the circular road as a square of four

road edges with equal properties. The junctions connecting the edges are configured such that vehicles do not slow down when approaching and while passing the junction. SUMO does not simulate the physical influence of curves, thus vehicles change their geographical direction instantaneously without decelerating.

SUMO normally removes vehicles from the road when they reach their destination edge. To avoid this, SUMO offers *rerouters*, that assign a new route to a vehicle once it reaches a certain edge. We apply rerouters to redirect vehicles on the bottom edge to the top edge and vice versa to keep them circling infinitely around the square.

All simulation runs in this chapter apply control scripts that start each simulation run with SUMO/Shawn and resume it with CircSim while keeping the model-parameters like length of road, number of lanes, number of vehicles, equipment rate, vehicle types including their fractions and properties. Only the current micro-state is reset such that all vehicles are stopped and repositioned to be equidistant. This prevents artificial effects caused by the last SUMO micro-state. For each simulator the simulation starts without Jam-ADS and turns it on after a while without changing anything else. Thus we have four different time slots: SUMO/Shawn without Jam-ADS, SUMO/Shawn with Jam-ADS, CircSim without Jam-ADS, and CircSim with Jam-ADS. In the plots, the time slot borders are marked by vertical or horizontal lines at the respective times.

To resume the SUMO/Shawn simulation with CircSim all simulation data are converted to a CircSim simulation state file, which is loaded into CircSim.[1] This means that all plots of a single simulation run[2] in this chapter are produced by CircSim. Particularly, the plotted fuel consumption of both SUMO/Shawn and CircSim time slots is computed completely by CircSim according to its fuel consumption model (Section 6.3.1) with the fuel consumption configuration given in Table 6.1.

As explained in Section 6.2.1 we gave the passenger cars a high maximum velocity and set the road's maximum velocity below that to prevent the mentioned reduction of randomness in SUMO due to approaching the maximum vehicle velocity. However, we assign to trucks (not present in single lane settings) a maximum velocity of 80 km/h, because we assume they cannot accelerate much more and have therefore less random deceleration. When the simulation runs switch to CircSim the different maximum velocities of the road and the passenger cars are kept like other configuration values, although this is not necessary for CircSim with its constant maximum acceleration.

[1] The other direction is not possible, because SUMO does not provide a facility to load a micro-state and resume simulation from there.

[2] This excludes the plots in Section 7.6 summarizing an entire simulation session

Table 7.1.: Configuration settings for single-lane simulations. These values are close to the values listed in Section 3.2.5 to generate traffic jams. They are not exactly the same, because these values belong to a large simulation session with many different model-parameter values. The values below the dashed line apply only if Jam-ADS is turned on.

Setting	Value
Road length	6000 m
Number of lanes	1
Density	50 vehicles/km/lane $\Leftrightarrow$ 300 vehicles
Maximum deceleration	4.5 m/s^2
Vehicle length	5 m
Maximum velocity of road	140 km/h
Maximum acceleration	1.5 m/s^2
Maximum velocity of vehicles	180 km/h
Equipment rate	99 %
Jam-ADS λ	0.67
Jam-ADS distance	766 m

7.3. Single lane

In our first experiments we computed the Jam-ADS average over a fixed number of vehicles ahead. Despite working well, this is not realistic, as in sparse traffic these vehicles may be to far away to receive their data and in dense traffic it would be better to average over more vehicles to give statistical outliers less influence. Thus, we decided to focus on averaging over a fixed distance ahead of each vehicle that we call *Jam-ADS distance*.

The configuration settings of single-lane simulation results presented in this section are given in Table 7.1.

7.3.1. With full equipment

To get a first impression how Jam-ADS works we present a simulation run on a single lane with (nearly) all vehicles equipped. The presented simulation run was part of a larger simulation session and this run was configured with 100 % equipment rate, but because of rounding errors 4 out of 300 vehicles remained unequipped. However, this helps to show some further properties.

The differences between SUMO/Shawn and CircSim visible in the following plots originate from the difference between the old (Equation (6.1)) and the recent (Equation (6.2)) expression for v_{safe} in SUMO. Replacing the recent with the old v_{safe} expression in SUMO equalizes the results of the SUMO/Shawn and

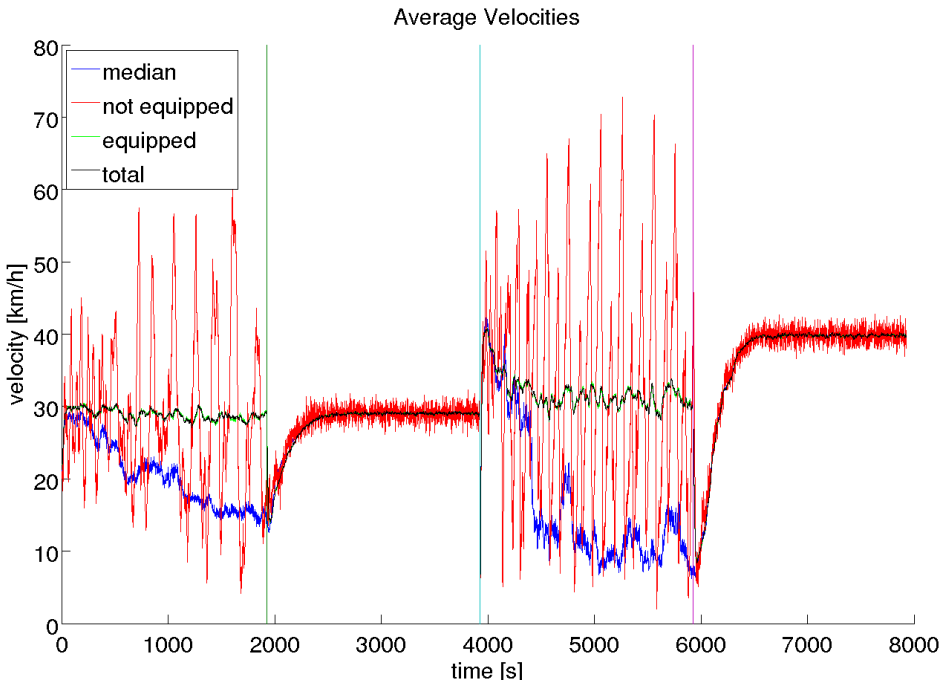


Figure 7.1.: Average velocities over time on a single lane with full equipment. The four time slots from left to right are SUMO/Shawn without Jam-ADS, SUMO/Shawn with Jam-ADS, CircSim without Jam-ADS, and CircSim with Jam-ADS.

the corresponding CircSim time slots up to random fluctuations. Note, that the difference between SUMO's Gipps-like v_{safe} (Equation (6.1)) and the original v_{safe} of the Krauß model (Equation (3.9)) is negligible according to our tests.

Nevertheless, after proving this, we stayed with the default v_{safe} expression in SUMO, because it is validated by the SUMO maintainers and many other users. Additionally, testing our strategy successfully with different models shows that it does not depend on artificial peculiarities of the traffic model.

We now look at the results in terms of the plot types described in Section 6.3.2; see there for general descriptions of the plots. We start with a look at the average velocity over time (Figure 7.1). As explained, the vertical lines mark the borders between the four time slots. As can be seen, each time slot starts with a settling phase until the system reaches a stationary state.

The first and third time slots (without Jam-ADS) show a median much smaller than the arithmetic average. This means that most of the vehicles are slower than

the arithmetic average and only a few vehicles are faster. With Jam-ADS in the second and fourth time slot the median is nearly exactly (second time slot) or close (fourth time slot) to the arithmetic average, thus Jam-ADS produces a more symmetric velocity distribution. Further plots presented next confirm this result.

The red line of the unequipped vehicles oscillates heavily, because it averages over only four vehicles, while the green line of equipped vehicles averages over 296 vehicles, which is (because of the law of large numbers) always very close to the black line of the average over all 300 vehicles. Without Jam-ADS, individual vehicles alternate between high speed in free traffic and nearly stopping in traffic jam. The average over the four unequipped vehicles reflects that fact. With Jam-ADS, oscillations of the red line are much smaller, which also shows the positive effect of Jam-ADS.

When turning Jam-ADS on at the start of the second and fourth time slots we observe the following typical behavior: The arithmetic average decreases dramatically before recovering again. This follows from the fact that Jam-ADS can only slow down individual vehicles but never make them move faster, because without Jam-ADS every vehicle already moves as fast as desired, which is as fast as possible with respect to its predecessor. Thus, Jam-ADS starts working by slowing down some vehicles that otherwise would be faster and it takes some time until the global effect of Jam-ADS sets in. In case of CircSim we also see a considerable increase of the arithmetic velocity average with Jam-ADS (fourth time slot) because of the mentioned different v_{safe} of SUMO and CircSim.

The velocity-distribution-development plot in Figure 7.2 supports the observations made at the average velocity plot. The four time slots are not marked but can be clearly distinguished from back to front. Note particularly, the reset of the micro-state at the start of CircSim at approximately 4000 seconds as zero-velocity peak.

It can be seen that without Jam-ADS we have a wide distribution of velocities with a maximum at zero (CircSim) or close to zero (SUMO/Shawn). Applying Jam-ADS narrows the distribution around the average. There are no more high-speed vehicles, but there are neither any low-speed vehicles.

Figure 7.3 contains a plot of the micro-states before and after turning Jam-ADS on for SUMO/Shawn and CircSim. The two subfigures are from the same plot but limited to different time ranges around the time slot borders where Jam-ADS is turned on. The complete plot is too large to be displayed here.

The upper parts of both plots (without Jam-ADS) show the development of several traffic jams with free traffic in between. In the larger free traffic areas, vehicles are able to reach nearly maximum velocity. Because of random deceleration they only move occasionally with exactly maximum velocity for a single simulation step.

The horizontal lines in each plot mark the moments when Jam-ADS is turned on. Vehicles upstream of traffic jams are slowed down such that the traffic jams

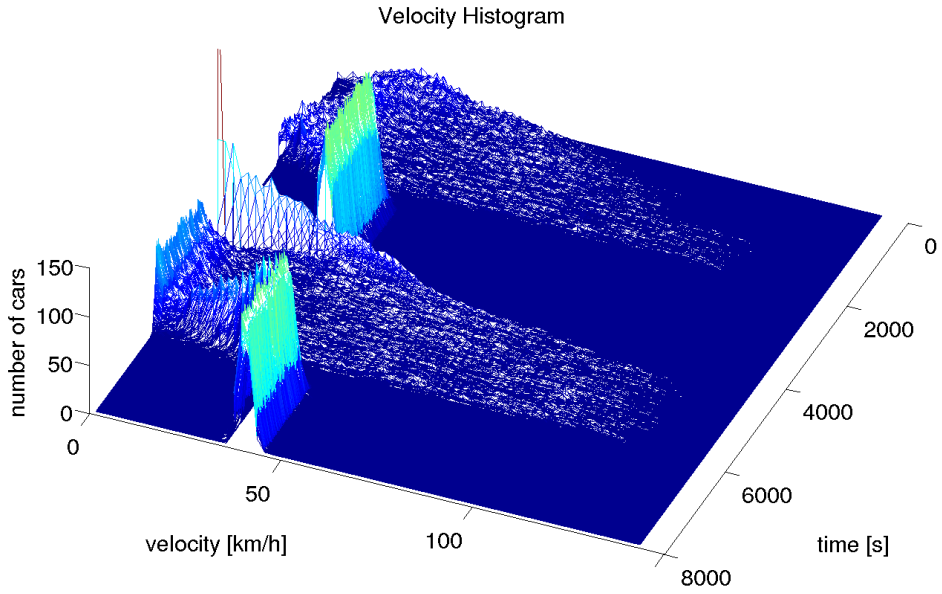
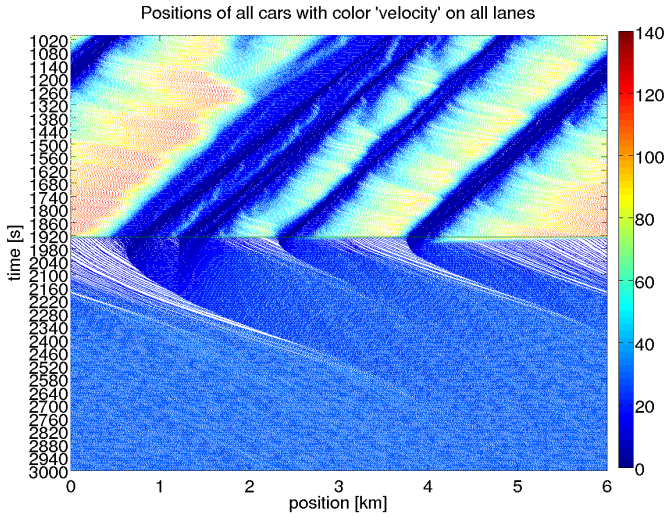
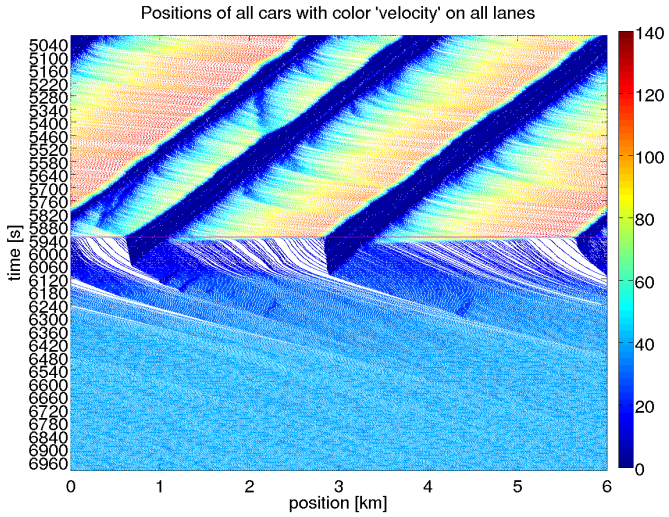


Figure 7.2.: Velocity distribution development on a single lane with full equipment.

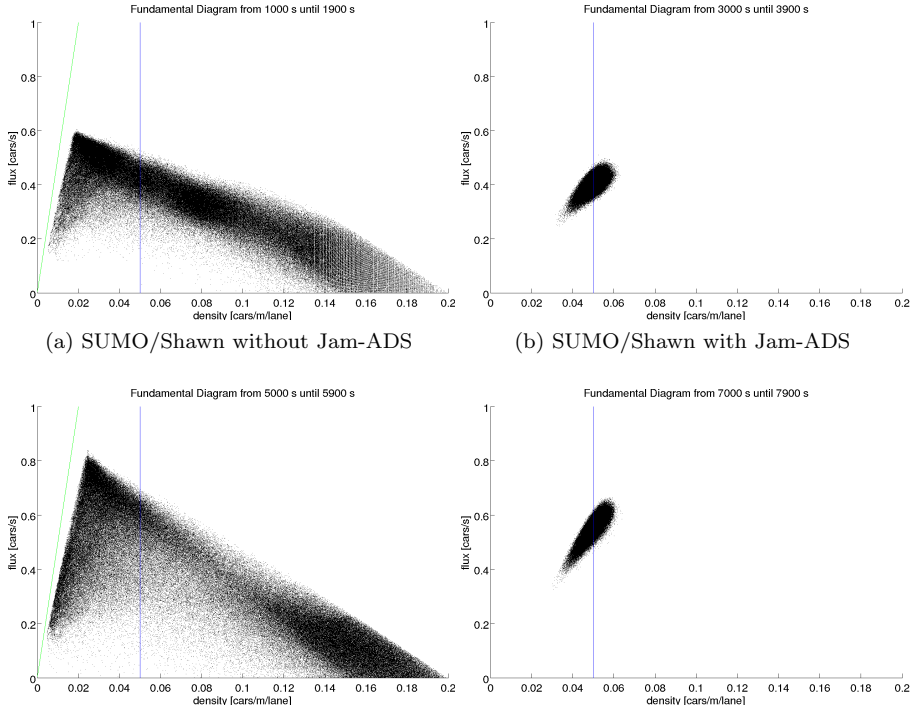


(a) SUMO/Shawn before and after turning Jam-ADS on



(b) CircSim before and after turning Jam-ADS on

Figure 7.3.: Positions plot of single lane with full equipment. Displayed are different time sections of a single plot.



(a) SUMO/Shawn without Jam-ADS

(b) SUMO/Shawn with Jam-ADS

(c) CircSim without Jam-ADS

(d) CircSim with Jam-ADS

Figure 7.4.: Fundamental diagrams of single lane with full equipment.

can dissolve. After dissolving the jams a new stationary state sets in, at which traffic flow becomes more uniform than without Jam-ADS.

Note, that traffic jams are dissolved from the upstream end. As explained in Section 2.2 it has to be assured to have smaller inflow into the upstream end of a traffic jam than current outflow at the downstream end, which Jam-ADS can enforce.

Figure 7.4 contains four fundamental diagram plots corresponding to the four time slots. The time ranges are chosen such that the non-stationary phases are excluded. See Section 2.1 for a general description of fundamental diagrams and Section 6.3.2 for the specific features of this plot type.

The two fundamental diagram plots without Jam-ADS (Figure 7.4a and Figure 7.4c) exhibit the typical triangle form of empirical fundamental diagrams (Section 2.1). We have a thin branch of free traffic and a wide branch of congestion. The different slopes of the jam branches resemble the different velocities with

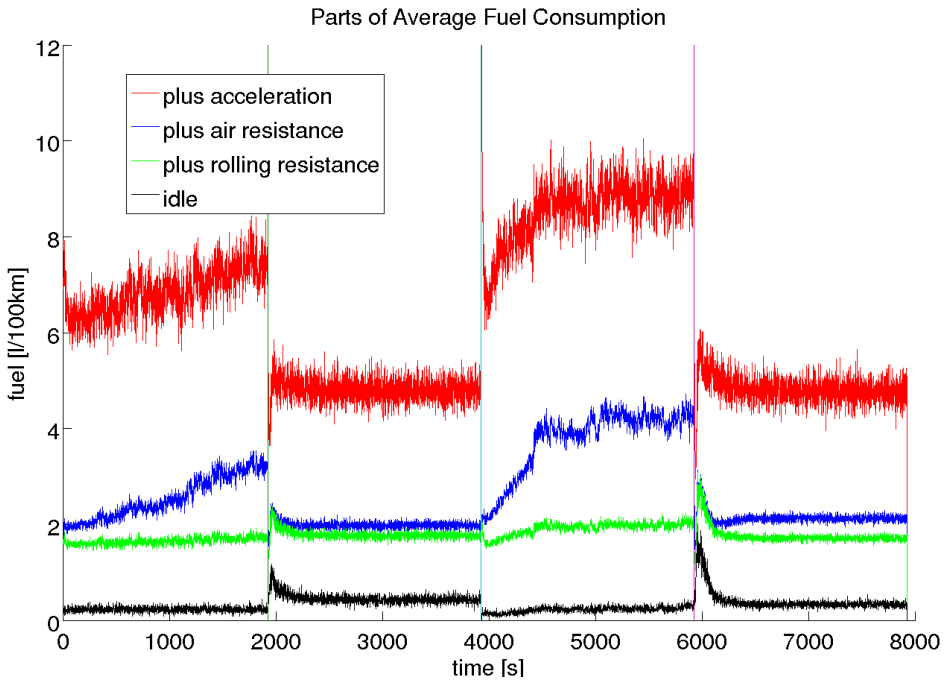


Figure 7.5.: Fuel consumption on a single lane with full equipment. The four time slots from left to right are SUMO/Shawn without Jam-ADS, SUMO/ Shawn with Jam-ADS, CircSim without Jam-ADS, and CircSim with Jam-ADS.

which jam fronts travel upstream in SUMO/Shawn and in CircSim. This is again an effect of the different v_{safe} expressions in SUMO and CircSim, as explained above.

The fundamental diagram plots with Jam-ADS (Figure 7.4b and Figure 7.4d) give another view on the effects of Jam-ADS. The jam branches and also the low-flow free traffic are completely eliminated.

Finally, we take a look at the average fuel consumption plot in Figure 7.5. With Jam-ADS (second and fourth time slot) we have a massive drop of the total average fuel consumption (red line).

The largest saving is due to the acceleration part, because the narrow velocity distribution with Jam-ADS needs a much smaller number of accelerations than without Jam-ADS. Also the air resistance part drops dramatically. The main contribution to the air resistance consumption comes from fast vehicles, because air resistance is proportional to the square of the velocity. With Jam-ADS we have

fewer high-speed vehicles, leading to smaller air resistance consumption. Rolling resistance consumption is proportional to the velocity, hence it remains about the same. The idle consumption is a little larger with Jam-ADS, because we have in total less deceleration, leading to more vehicles falling into the fuel cut-off deceleration range, which causes idle consumption (see Section 6.3.1).

7.3.2. Limited equipment rate

In the last section, we presented the effect of Jam-ADS if (nearly) all vehicles are equipped. However, it is not reasonable that all vehicles are equipped with Jam-ADS devices and that all drivers always comply with their velocity recommendations. Especially during market introduction only few vehicles are equipped, which leads to the question how well the strategy works with lower equipment rates.

To show that Jam-ADS also works with low equipment rates we repeat the same simulation run as in the last section, but now we limit the equipment rate to 5 %, which means only 20 out of the 300 vehicles are equipped. We need to compensate this by a longer Jam-ADS distance of 1500 m to include a reasonable number of vehicles in the Jam-ADS average. Figures 7.6 to 7.10 contain the resulting plots corresponding to above plots with full equipment. Generally, the limited equipment rate makes no visible difference with SUMO/Shawn, but with CircSim the difference is more noticeable.

The total average velocity plot (Figure 7.6) of CircSim with Jam-ADS shows more oscillations than without Jam-ADS, but the median is still close to the average in contrast to the time slot without Jam-ADS. In this run we have in SUMO/Shawn an increase of the average velocity because of Jam-ADS.

The velocity distribution plot (Figure 7.7) also exhibits a clear narrowing in SUMO/Shawn, but a more broad distribution in CircSim, although it is still better than without Jam-ADS.

The fundamental diagram plots (Figure 7.8d) of SUMO/Shawn with Jam-ADS are also nearly as good as with full equipment, but under CircSim the jam branch and the lower part of the free traffic branch are not completely eliminated by Jam-ADS. But, as the color distribution shows, most vehicles outside of the top corner are non-equipped vehicles. Thus, there are clear benefits for equipped vehicles even at this low equipment rate.

At least, the effect on fuel consumption (Figure 7.9) is nearly as good as with full equipment. Jam-ADS still reduces fuel consumption dramatically with the same effect on the four consumption parts as with full equipment rate.

The positions plot of SUMO/Shawn (Figure 7.10a) and CircSim (Figure 7.10b) show the difference between SUMO/Shawn and CircSim both using Jam-ADS. SUMO/Shawn produces, even with only 5 % equipment rate, a uniform traffic flow, while under CircSim several small traffic jams arise and are dissolved, soon afterwards.

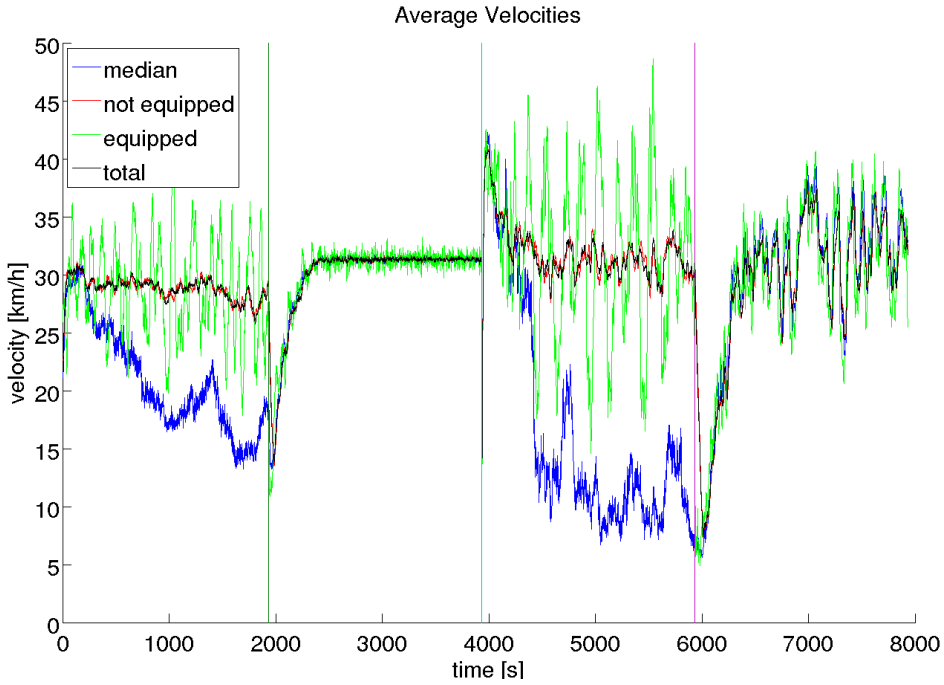


Figure 7.6.: Average velocities over time on a single lane with 5 % equipment.

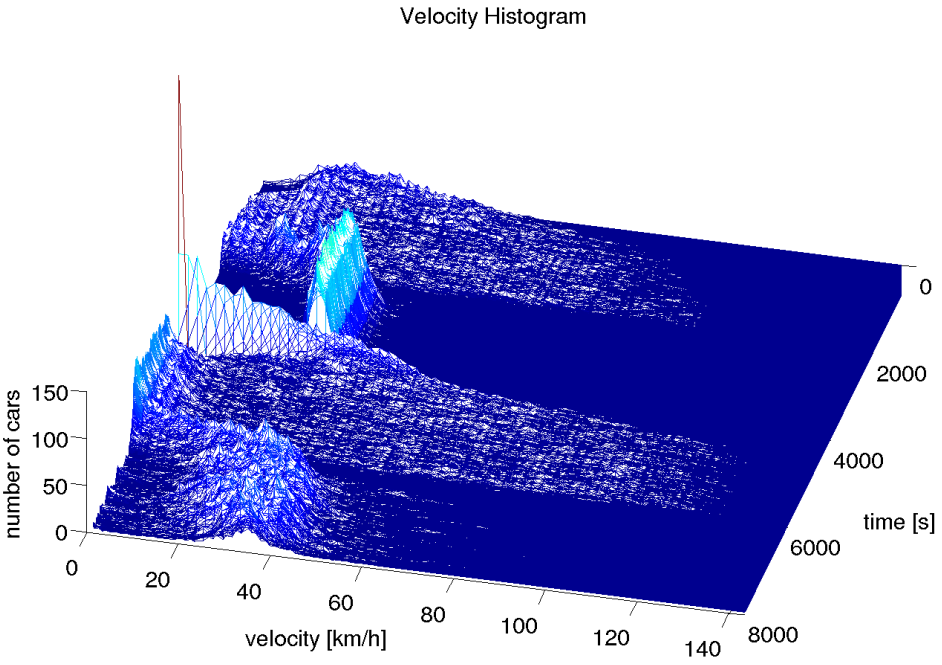
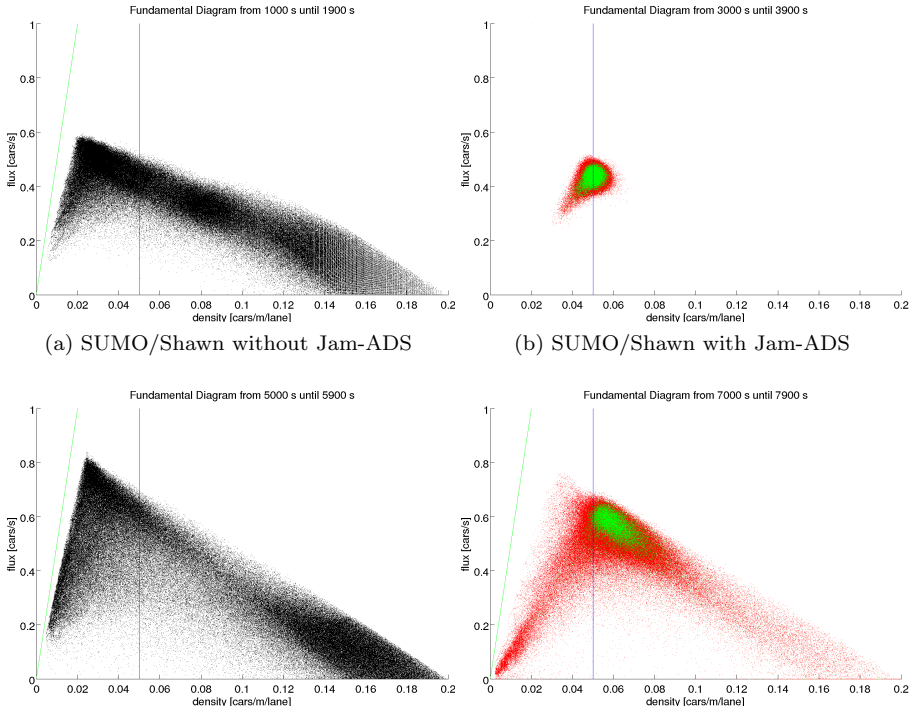


Figure 7.7.: Velocity distribution development on a single lane with 5 % equipment.



(a) SUMO/Shawn without Jam-ADS

(b) SUMO/Shawn with Jam-ADS

(c) CircSim without Jam-ADS

(d) CircSim with Jam-ADS

Figure 7.8.: Fundamental diagrams of a single lane with 5 % equipment. In the right plots, green data points belong to equipped vehicles and red data points to non-equipped ones. The numbers of visible green and red data points do not resemble the actual equipment rate, because some points are drawn on top of each other, particularly green points on top of red points.

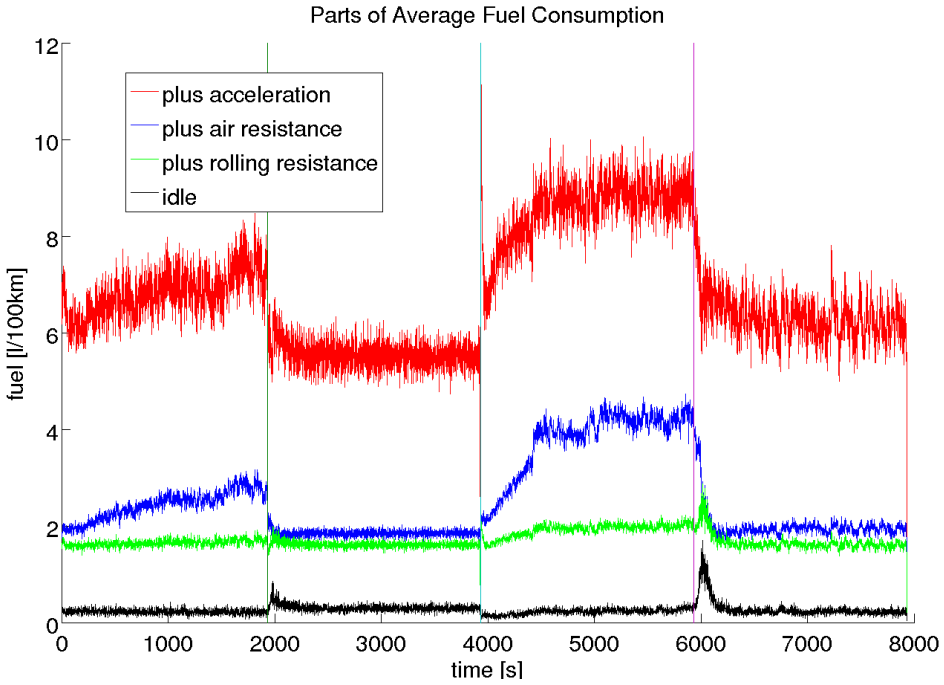
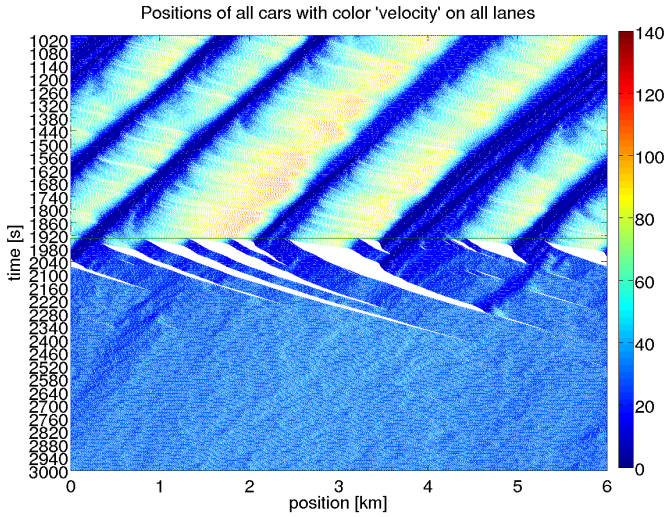
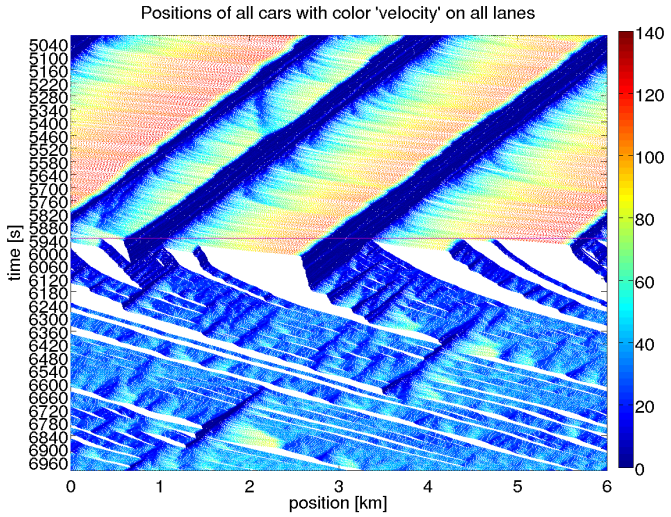


Figure 7.9.: Fuel consumption on a single lane with 5 % equipment.



(a) SUMO/Shawn before and after turning Jam-ADS on



(b) CircSim before and after turning Jam-ADS on

Figure 7.10.: Positions plot of a single lane with 5 % equipment. Displayed are different time sections of a single plot.

The gaps in the positions-plot time-slots with Jam-ADS occur in front of equipped vehicles whenever they approach a traffic jam and all equipped vehicles they can "see" are within that traffic jam. Thus, they adapt their velocity to the slow equipped vehicles within the jam, but the non-equipped vehicles between any such equipped vehicle and the next traffic jam move as fast as possible, opening the gap behind.

With SUMO/Shawn this effect is sufficient to completely dissolve all traffic jams and prevent new ones to emerge, but this is not the case with CircSim. The reason is that CircSim produces slightly different traffic jams because of the mentioned difference in v_{safe} that allows higher velocities in CircSim when following another vehicle. Traffic jams in CircSim are denser and travel upstream faster than those in SUMO/Shawn. The higher number of vehicles bound in CircSim's traffic jams leaves more space for the remaining vehicles to move faster.

However, the simulations with two independent simulators show that even a low equipment rate is sufficient to improve traffic flow with Jam-ADS on a single lane.

7.4. Multiple lanes

We have seen that Jam-ADS works well on a single-lane road even with a low equipment rate. This is not really surprising, because when an equipped vehicle slows down by following the recommendations of Jam-ADS unequipped vehicles behind are forced to slow down as well. But with multiple lanes the situation is fundamentally different. Unequipped vehicles could pass equipped ones, which would make their deceleration due to Jam-ADS completely useless. This section investigates multiple-lane roads with limited equipment rate.

To make the simulations with multiple lanes more realistic, we add trucks as additional vehicle type. Trucks have a smaller maximum velocity and acceleration, and are longer than passenger cars. As explained in Section 6.3.1 we apply the same fuel consumption settings for each vehicle type as given in Table 6.1.[3] All specific configuration settings of multiple-lane simulations presented in this section are given in Table 7.2.

The first simulation tests (not presented here) showed that the Jam-ADS average should only be taken over the current lane. Otherwise, passenger cars on the left lane, passing trucks on the right lane, are forced to adapt their velocity to the smaller velocity of the trucks even if there is no need to slow down. Thus, non-lane-wise Jam-ADS would unnecessarily reduce the flow of free traffic.

[3] Note, that CircSim computes the fuel consumption of the closed system for both SUMO/Shawn time slots and for CircSim time slots as explained in Section 7.2.

Table 7.2.: Configuration settings for multiple lane plots.

Setting	Value
Road length	6000 m
Probability of random lane change	0.02
Density	41 veh/km/lane
Truck fraction	10 %
Maximum deceleration	4.5 m/s^2
Maximum velocity of road	140 km/h
Maximum velocity difference on the lanes (Section 6.3.1)	1 m/s
Passenger car length	5 m
Maximum acceleration of passenger cars	1.5 m/s^2
Maximum velocity of passenger cars	180 km/h
Truck length	12 m
Maximum acceleration of trucks	0.4 m/s^2
Maximum velocity of trucks	80 km/h
Jam-ADS λ	0.67
Jam-ADS distance	3000 m

7.4.1. Two lanes

As an example of a two-lane simulation we present a run with approximately 5 % equipment rate; 32 of the 496 vehicles are equipped.

Figure 7.11 shows the development of the average velocity with the usual four time slots. We can see the same effects as seen with the single-lane simulation run. When Jam-ADS is turned on, the average velocity slightly increases and the median becomes nearly the same as the arithmetic average. The few equipped vehicles (green line) exhibit higher fluctuations than the other averages, because their number is much smaller than that of equipped vehicles.

Figure 7.12 contains the development of the velocity distribution. It shows again a narrowing of the distribution around the average value due to Jam-ADS, although in SUMO/Shawn it is less sharp than in the single-lane case. With CircSim we see with Jam-ADS fewer extremes than in the single-lane case.

Figure 7.13 shows the fundamental-diagram plots of the four time slots. They are colored according to vehicle type. Here we can see two pairs of straight lines arising from the origin marking the different maximum velocities of vehicle types on each lane.[4] We can also see that the jam branches of trucks and passenger cars are quite different (see Section 6.3.2 for explanation). Like in the single-lane run Jam-ADS leads to a concentration of the data points around the maximum flow.

[4]Section 6.3.1 explains why neighboring lanes have a slightly different maximum velocity. The difference in this case is denoted as "maximum velocity difference on the lanes" in Table 7.2.

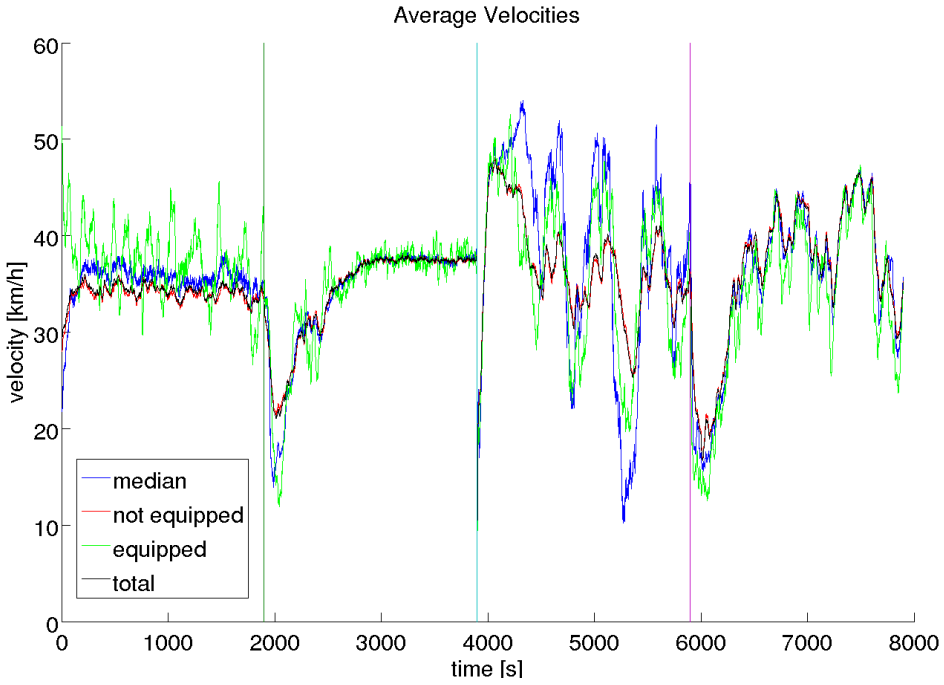


Figure 7.11.: Average velocities over time on two lanes with 5 % equipment rate.

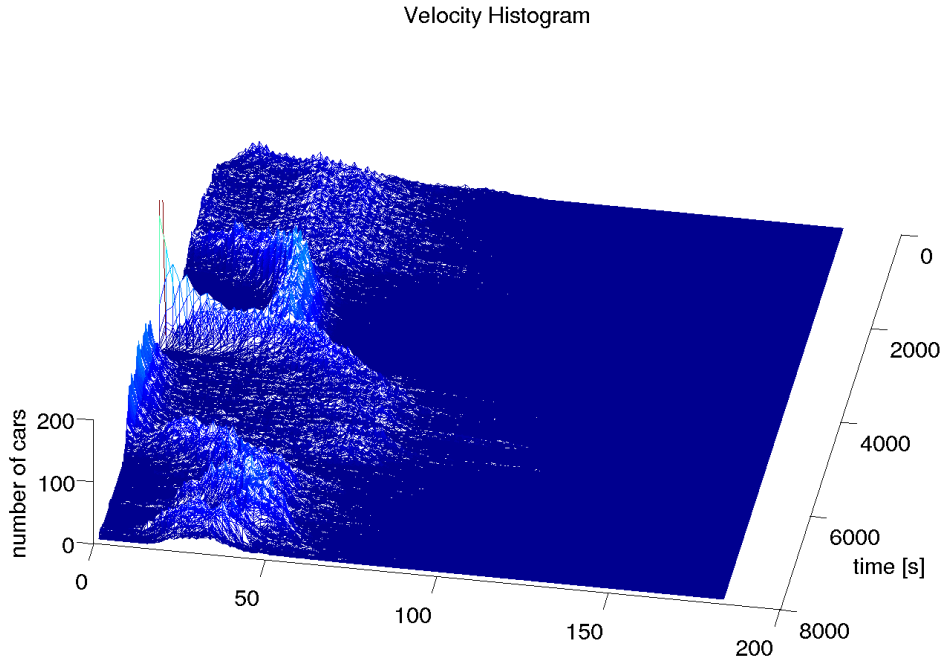
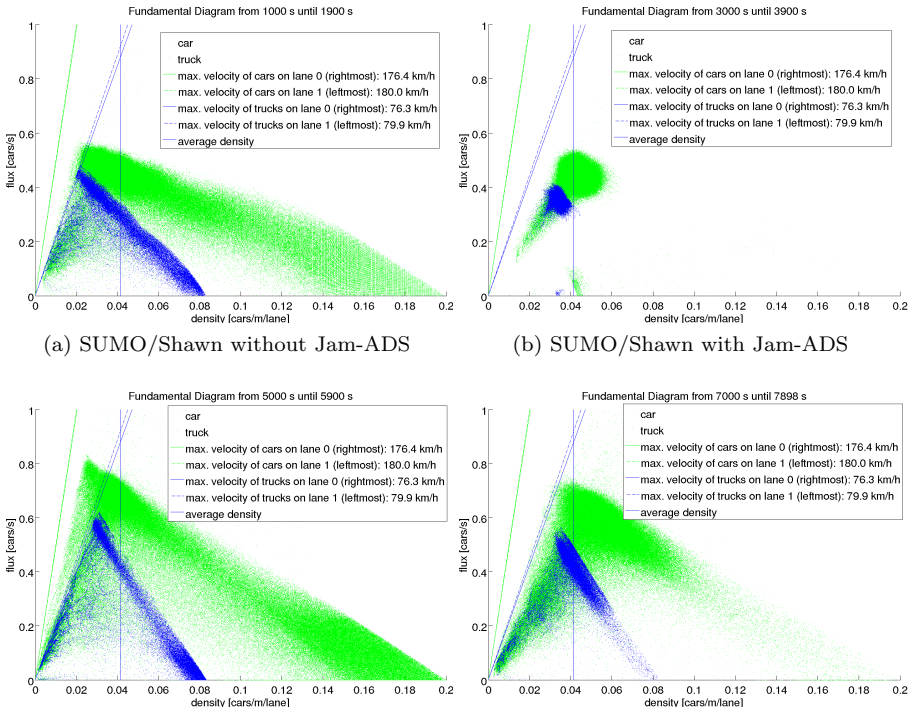


Figure 7.12.: Velocity distribution development on two lanes with 5 % equipment rate.



(a) SUMO/Shawn without Jam-ADS

(b) SUMO/Shawn with Jam-ADS

(c) CircSim without Jam-ADS

(d) CircSim with Jam-ADS

Figure 7.13.: Fundamental diagrams on two lanes with 5 % equipment rate. The color denotes the vehicle type: green for passenger cars and blue for trucks. The legends explain the meanings of the straight lines as explained in Section 6.3.2.

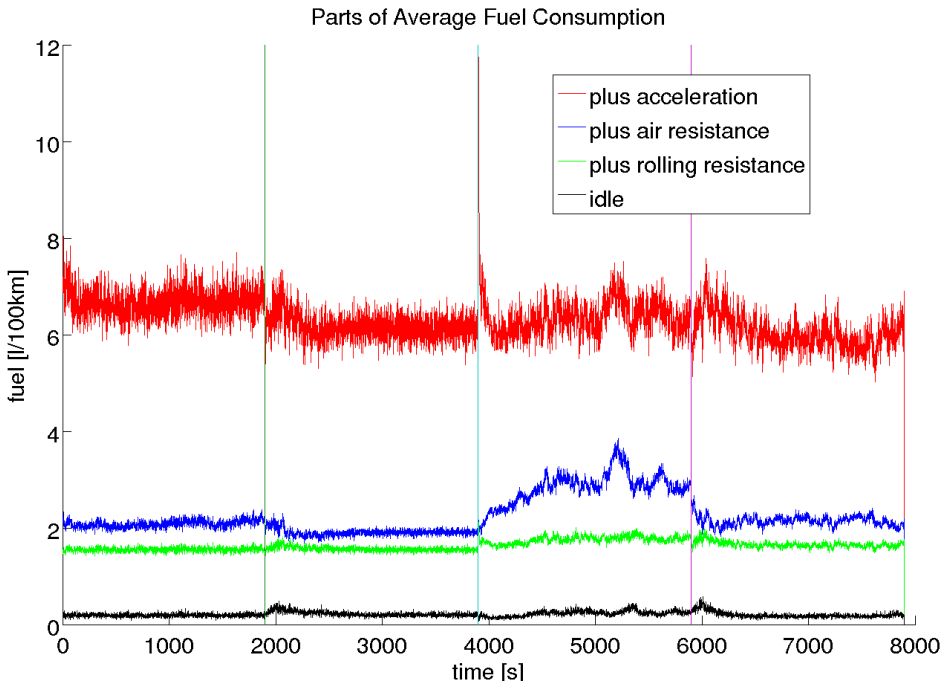
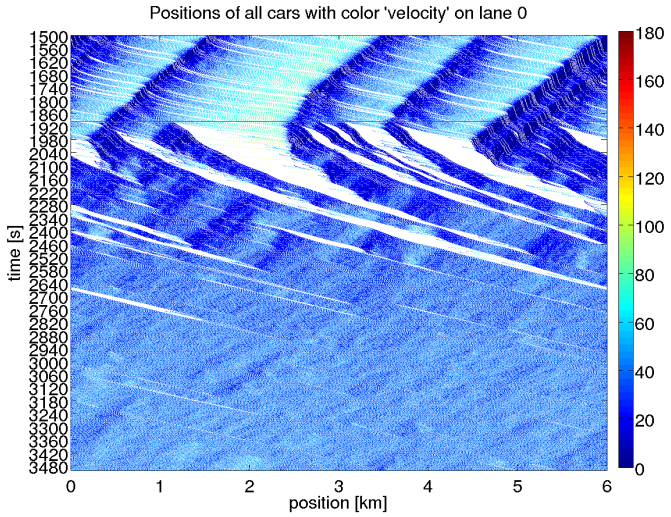


Figure 7.14.: Fuel consumption on two lanes with 5 % equipment rate.

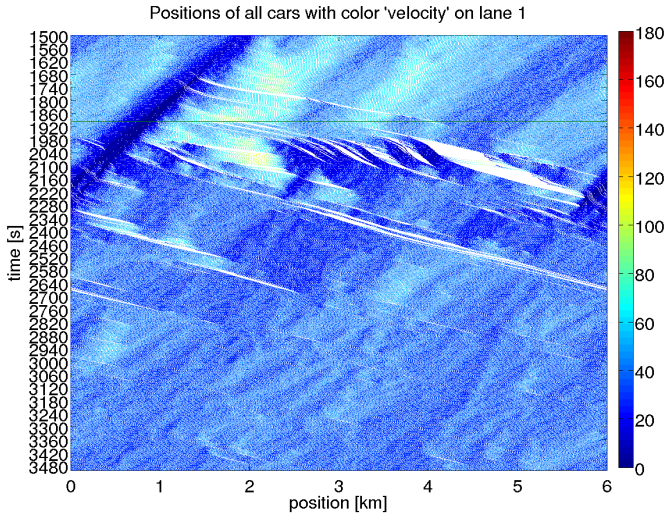
In Figure 7.14 we present fuel consumption. The fuel savings with Jam-ADS are less dramatic as for a single lane with 5 % equipment rate, but still considerable with approximately one liter per 100 km. Higher equipment rates show savings of up to two liters per 100 km.

Figures 7.15 and 7.16 show for SUMO/Shawn and CircSim, respectively, the position plots before and after turning Jam-ADS on. Like the single-lane simulation run with low equipment rate we can again see that the few equipped vehicles slow down the others such that traffic jams become dissolved (at least mainly dissolved under CircSim) and the start of new traffic jams is suppressed.

A closer investigation of the positions plots reveals why Jam-ADS works with low equipment rates on multiple lanes, as opposed to expectations. When an equipped vehicle on the right lane approaches a traffic jam and slows down following the Jam-ADS recommendation, it is at first passed by unequipped vehicles. Sooner or later an equipped vehicle from upstream starts to pass as well, but has to decelerate when it is level with the equipped vehicle on the right, because it follows the same velocity recommendation. From then on both vehicles slow down upstream traffic, which has the same effect as on a single-lane road.

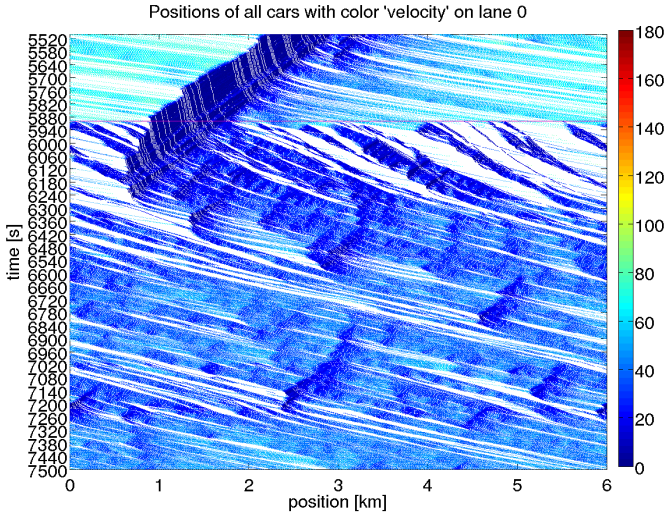


(a) Right lane of SUMO/Shawn before and after turning Jam-ADS on

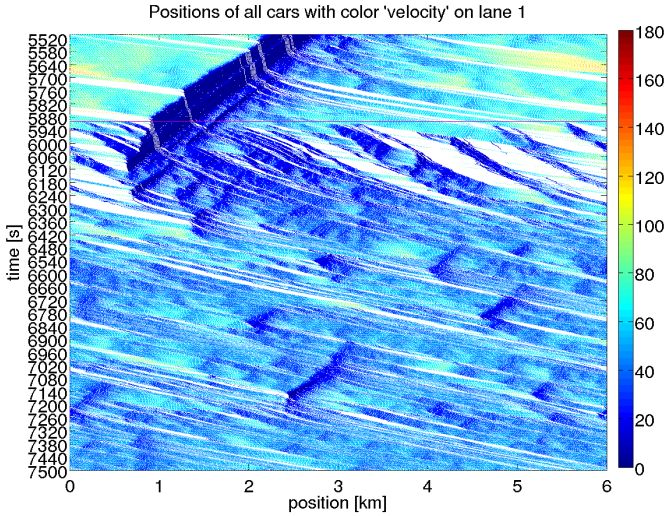


(b) Left lane of SUMO/Shawn before and after turning Jam-ADS on

Figure 7.15.: Positions plots of SUMO/Shawn on two lanes with 5 % equipment rate.



(a) Right lane of CircSim before and after turning Jam-ADS on



(b) Left lane of CircSim before and after turning Jam-ADS on

Figure 7.16.: Positions plots of CircSim on two lanes with 5 % equipment rate.

Although Jam-ADS works on two-lane roads, there are two reasons why it works less good. First, it takes some time until two equipped vehicles meet on both lanes as described. Second, each vehicle only considers equipped ones that are ahead on its own lane as explained above. Thus, each vehicle may have a different velocity recommendation by Jam-ADS, such that the corporate effect is smaller.

Finally, in Figure 7.17 we look at the plot of the number of lane changes for different equipment rates. It can be clearly seen, that SUMO has a completely different lane change model (as described in Section 6.2.1) than CircSim (as described in Section 6.3.1), such that CircSim performs many more lane changes than SUMO.

The plots show that, at least under CircSim, Jam-ADS reduces the number of lane changes at higher equipment rates, although there was no direct influence on lane changing implemented in Jam-ADS for these simulation runs. The narrow velocity distribution under Jam-ADS only reduces the necessity to switch lanes.

The striking difference of the number of lane changes in SUMO and CircSim does not follow from the different v_{safe} in SUMO. Replacing the default v_{safe} expression according to Equation (6.2) by the older expression in Equation (6.1) does increase the number of lane changes slightly[5], but it is still far from the number of lane changes in CircSim. At least we have shown that also with SUMO/Shawn, Jam-ADS reduces the number of lane changes. Nevertheless, the main reason for the different number of lane changes are the completely different lane-change models in SUMO and CircSim.

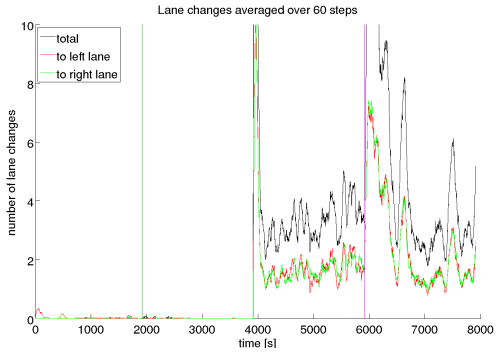
Further simulations revealed that Jam-ADS works best with a value of λ between 1/2 and 2/3, which is in accordance with the theoretical results of Chapter 9. The average velocity grows with growing λ as long as λ is small enough to make Jam-ADS effective at all. When λ approaches one, the influence of Jam-ADS vanishes completely and the probability of congestion causing perturbations increases. Thus, a large λ produces a larger average velocity but has a higher probability of traffic jam. It is conceivable to enable the driver to select at his/her Jam-ADS device how much influence he/she will give Jam-ADS by manipulating some control to change λ.

We also found that the best length of the Jam-ADS distance depends on the equipment rate. A lower equipment rate requires a longer distance, but for higher equipment rates a shorter Jam-ADS distance produces better results.

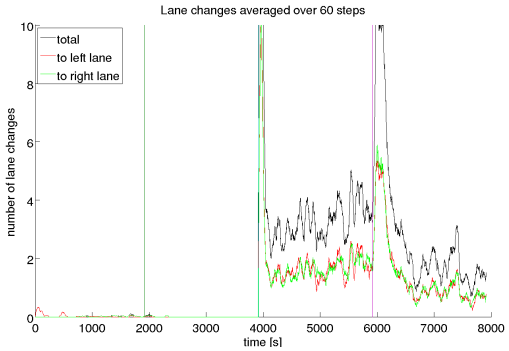
7.4.2. Three lanes

We now add a third lane to our simulations, but keep the other configuration values the same, except that we now have 168 equipped vehicles of 736 vehicles in

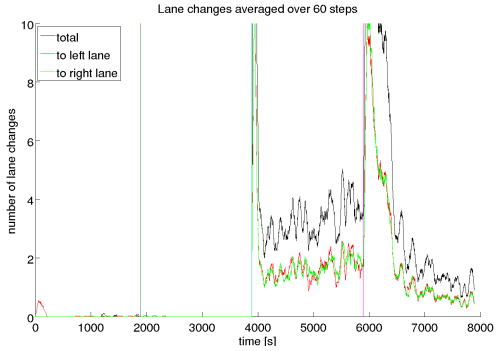
[5] It is increased, because a less strict v_{safe} is more likely to fulfill the safety conditions of lane changes.



(a) 5 % equipment rate



(b) 23 % equipement rate



(c) 99 % equipement rate

Figure 7.17.: Number of lane changes of several two-lane simulation runs with different Jam-ADS equipment rates.

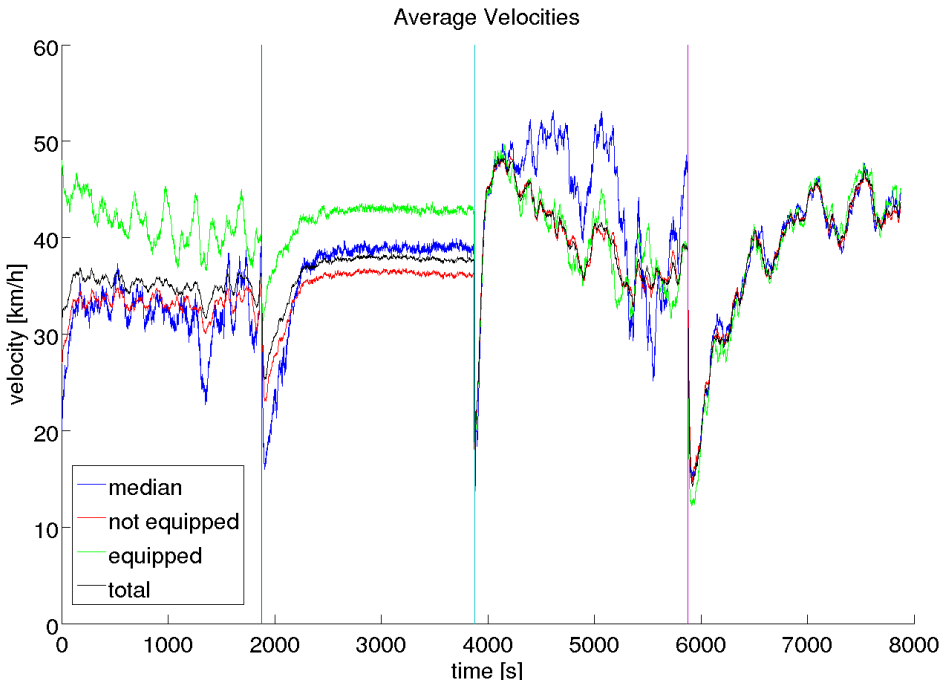


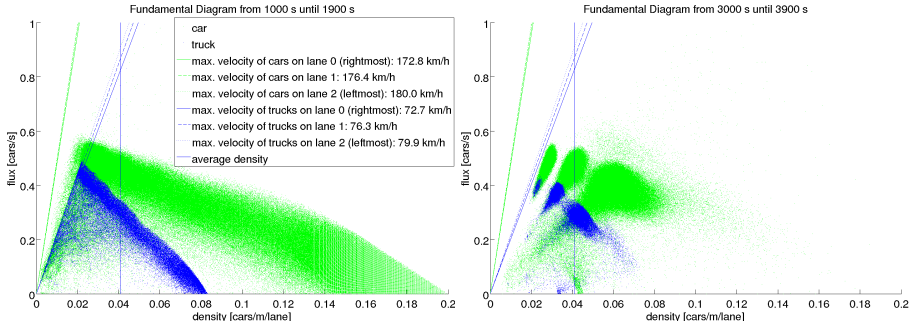
Figure 7.18.: Average velocities over time on three lanes with 23 % equipment rate.

total, which is an equipment rate of 23 %. With 5 % equipment rate the effects of Jam-ADS are hardly visible on three lanes.

Figure 7.18 contains the usual average-velocity-development plot for this simulation run. As before, we have an increase of the average velocity with Jam-ADS. The different average velocity of equipped and unequipped vehicles in SUMO/Shawn with Jam-ADS comes from a coincidentally larger fraction of equipped vehicles in the fast left lane than in the other lanes.

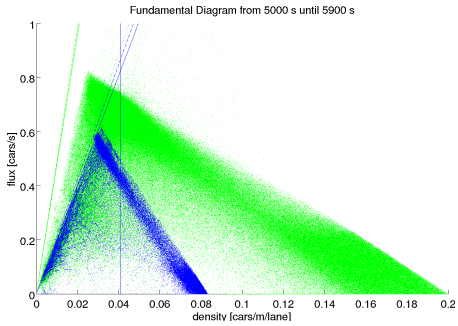
In Figure 7.19 we have the fundamental-diagram plots for the four time slots of SUMO/Shawn and CircSim without and with Jam-ADS, colored by vehicle type. Like the two-lane simulation run they exhibit the artificially distinct branches for passenger cars and trucks explained in Section 6.3.2.

The fundamental diagrams display basically the same effect of Jam-ADS as the other simulation runs above. But contrary to the two-lane simulation run we can see three different club-shaped aggregation areas at the plot of SUMO/Shawn with Jam-ADS (Figure 7.19b) belonging to the three lanes. Surprisingly, the club with the smallest velocity (the right one) corresponds to the middle lane while the one in the middle corresponds to the right lane. Only the small club on the left with

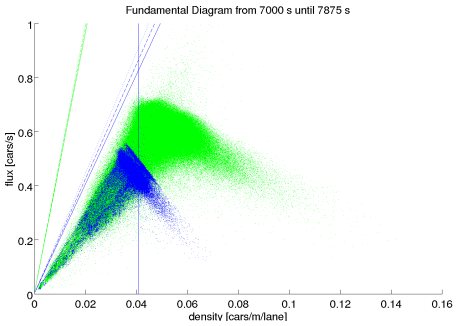


(a) SUMO/Shawn without Jam-ADS

(b) SUMO/Shawn with Jam-ADS



(c) CircSim without Jam-ADS



(d) CircSim with Jam-ADS

Figure 7.19.: Fundamental diagrams on three lanes with 23 % equipment rate.

the highest velocities corresponds to the left lane, as expected.

Observing the vehicles in the watch plot when Jam-ADS is turned on in SUMO/Shawn reveals what happens.[6] At the first moment after Jam-ADS is turned on the equipped vehicles on each lane decelerate and the unequipped vehicles start to queue behind the equipped ones forming dense platoons with large gaps between them. Those gaps can be observed with one or two lanes as well (see Figures 7.10 and 7.15). But in this simulation run, unequipped vehicles from the right lane switch into the gaps of the middle lane increasing the vehicle density in the middle lane. Soon after, Jam-ADS has reached equilibrium such that in each lane vehicles move with a common speed and are approximately equidistant. Because of the higher density in the middle lane its equilibrium velocity is smaller than the equilibrium velocity of the right lane. Note that, SUMO allows passing on the right if the velocity is below a jam threshold of 60 km/h.

Figure 7.20 confirms the higher density on the middle lane with SUMO/Shawn. The number of vehicles in the leftmost lane remains constant after the system has settled. Without Jam-ADS much more vehicles switch from the right to the middle lane. After Jam-ADS has settled all vehicles keep to their lane in SUMO/Shawn. Under CircSim there is much more permeability of the lanes, leading to higher oscillations of the number of vehicles in each lane, but also to less density differences.

Looking at the fuel consumption plot of the three-lane simulation run in Figure 7.21, we see the usual benefits of Jam-ADS.

7.5. Other variants

We tried several variants and extensions to improve Jam-ADS for the closed system. In this section we explain the most promising variants.

7.5.1. Lane recommendation

The data gained to recommend a velocity to the driver may be applied to derive a lane recommendation as well. It is particularly helpful that we have to average over only the vehicles on the current lane to derive the velocity recommendation. Thus, we already have the necessary lane-wise data.

Lane changing is a perturbation that may cause congestion if the local density is too high. So we focused on strategies to avoid lane changes if they are not beneficial for the driver. To do this we considered two different approaches.

The first variant recommends to stay on the current lane if the vehicle is inside a traffic jam. We assume to be inside a traffic jam if the vehicle's velocity is below

[6] Using CircSim's watch plot to investigate micro-states produced with SUMO/Shawn is possible after the conversion to a CircSim simulation state. See Section 6.4.1.

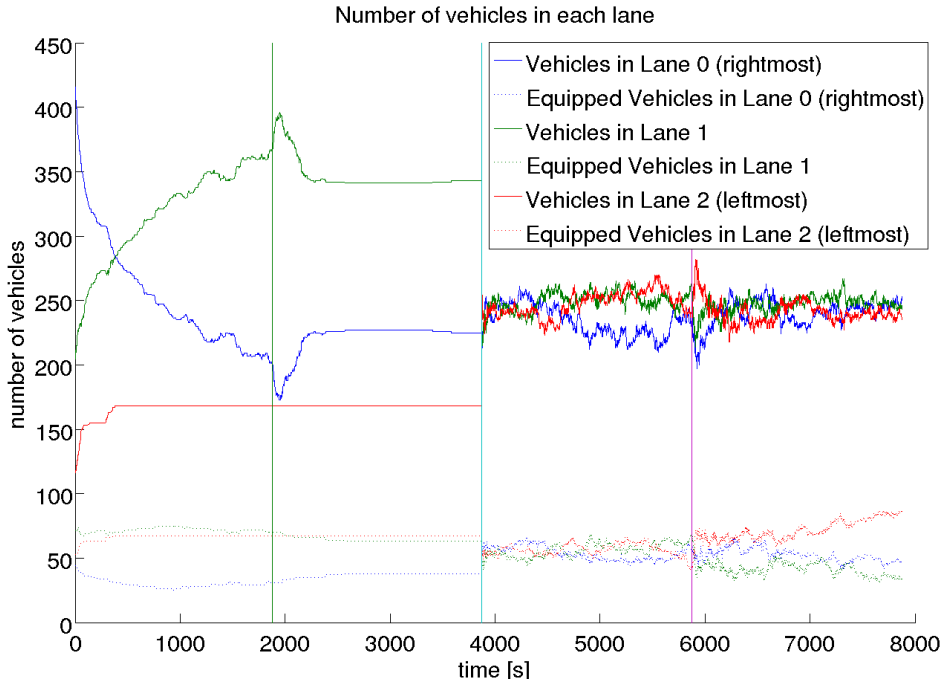


Figure 7.20.: Number of vehicles in each of three lanes with 23 % equipment rate.

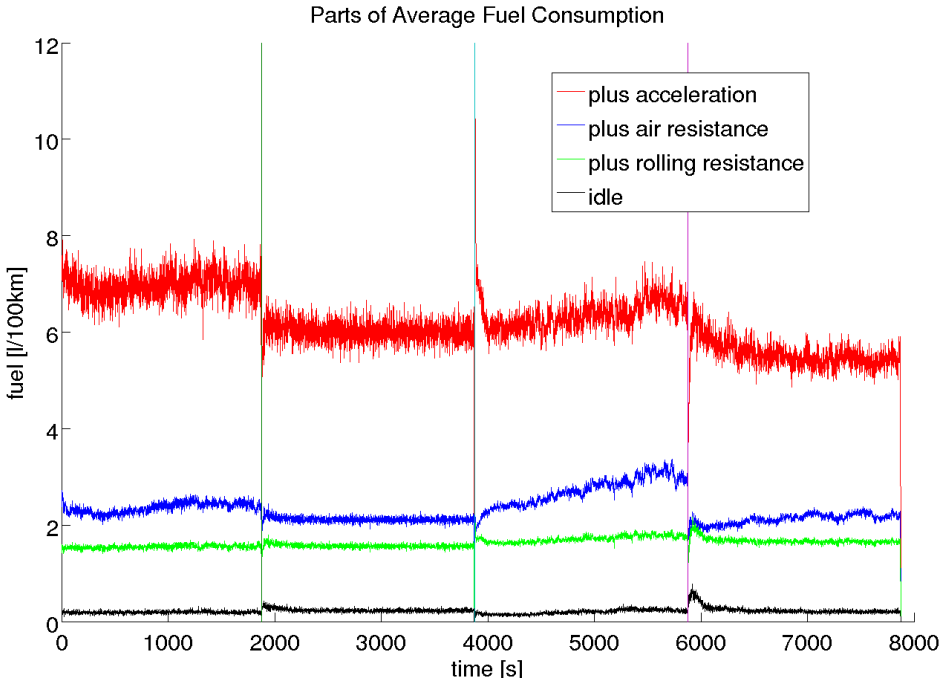


Figure 7.21.: Fuel consumption on three lanes with 23 % equipment rate.

a jam threshold. The second variant recommends to stay on the current lane if the average velocity on the preferred lane is only slightly different to the average velocity on the current lane, that is, the relation of both average velocities is below a threshold.

Tests of both strategies with different thresholds showed only negligible improvements of traffic flow. So we did not incorporate lane change recommendations into Jam-ADS.

7.5.2. Considering acceleration

Jam-ADS computes the average of the current velocities of the vehicles ahead. However, we actually want to know the average velocity ahead at the time we arrive at the position where the vehicles ahead are currently.

To better approximate this, we tested a variant of Jam-ADS that considers the average acceleration of the equipped vehicles ahead to estimate their future average velocity. As their single location we assume the median position of the equipped vehicles ahead that we average over. Assuming further that our vehicle keeps its velocity constant, which allows us to compute an approximate time t_{dist} until we arrive at the median position of the vehicles ahead.

To reduce the effect of short-term fluctuations, the average acceleration is also averaged over a given number of simulation steps. With Δt_a as the time interval corresponding to the number of simulation steps averaged over, $v_i(t)$ as velocity of vehicle i at time t, and $\{\text{vehicles}\}$ as the set of equipped vehicles to average over, the average acceleration a_{avg} is computed as

$$a_{\text{avg}} = \frac{1}{\Delta t_a |\{\text{vehicles}\}|} \sum_{i \in \{\text{vehicles}\}} v_i(t) - v_i(t - \Delta t_a) \quad . \tag{7.1}$$

Finally, t_{dist}, the mean acceleration a_{avg} of the equipped vehicles ahead, and a consideration factor f_{acc} updates v_{avg} according to

$$v_{\text{avg}} \leftarrow v_{\text{avg}} + f_{\text{acc}}\, a_{\text{avg}}\, t_{\text{dist}} \quad . \tag{7.2}$$

We tested this variant with different factors f_{acc} and intervals Δt_a. In the best cases the total average velocity was as good as the standard strategy without this extension. But in many cases we observed large oscillations of the total average velocity between high speeds and nearly standing. Thus, this approach turned out to be counterproductive.

7.6. Parameter dependency of Jam-ADS

During the research for this thesis numerous simulation runs produced a large number of plots. To be able to examine them in a systematic way, we aggregated

the data of complete simulation sessions into *summary plots*. The purpose of these summary plots is to discover which values of model-parameters[7] are better or worse.

The summary plots (Figures 7.22 to 7.25) show average velocity and fuel consumption over a model-parameter. Each data point is the aggregation of one time slot of one simulation run. The aggregation of the fuel consumption considers the peculiarities of averaging over a time range as explained in Section 6.4.4. Data points of simulation runs are connected by lines if their model-parameter sets only differ in the value assigned to the x-axis. Data points and their connecting lines are colored according to the time slot they belong to:

Blue SUMO/Shawn without Jam-ADS

Green SUMO/Shawn with Jam-ADS

Black CircSim without Jam-ADS

Red CircSim with Jam-ADS

The data aggregation for a data point is done over only a part of each time slot, which is a configurable time length at the end of each time slot, during which we assume the system to be in a stationary state. All examples in this chapter have time slots with a length of 2000 seconds and the averages of the summary plots are taken over the last 1000 seconds. The average of the fuel consumption is computed according to Equation (6.9) with the configuration values given in Table 6.1. The remaining configuration settings are the same as of the multiple-lane simulation runs above (Table 7.2), because the results presented above in this chapter belong to the simulation session whose summarize plots we present in the following.

Two simulation runs whose settings only differ in the random seed should expose the same averages regardless of different development of their micro-states. Figure 7.22 contains a summary plot with the random seed as independent model-parameter. Lines connect simulation runs that only differ by their random seed.[8] The plot shows that in most cases the expectation of random seed independence holds, but for some runs this is not the case, because the actual random seed causes a deviating behavior.

All other summary plots should be interpreted with this "random-seed dependency" in mind, particularly as the following summary plots belong to the same simulation session, thus they all contain the exact same set of average velocity and fuel consumption values only differing by their x-axis value and their connecting lines.

[7] See Section 6.1 for the definition of "model-parameter".

[8] The entire simulation session repeated every simulation run with the random seeds 1234 and 2345.

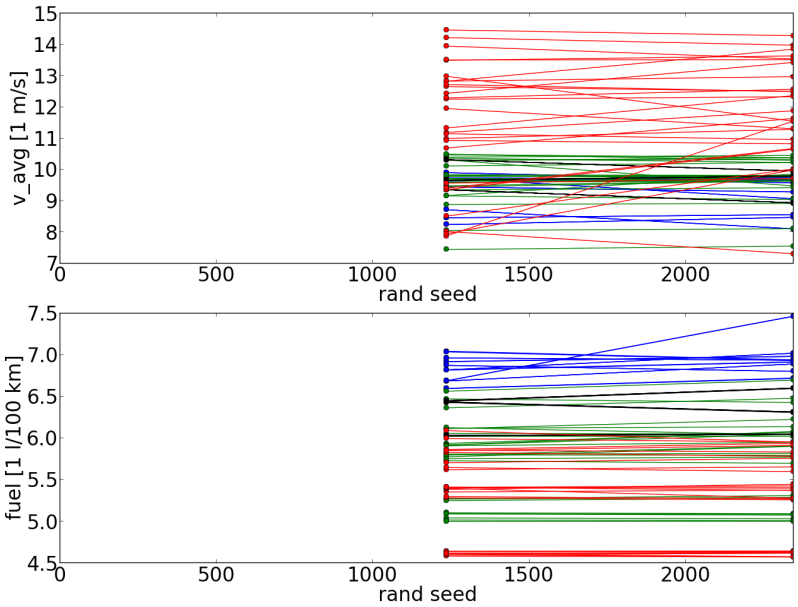


Figure 7.22.: Changes of averages with two different random seeds 1234 and 2345.

7.6.1. Equipment rate dependency

Figure 7.23 contains a summary plot depending on the equipment rate. Theoretically the blue and black lines without Jam-ADS should be exactly horizontal, because the behavior without Jam-ADS does not depend on the equipment rate. As can be seen in the summary plot, this is not always the case with SUMO/Shawn. The reason is that for SUMO we encode the equipment state of each vehicle into its ID (which is a string in SUMO) by adding a special prefix. In some cases when SUMO iterates over the vehicles the IDs determine the exact order, thus different IDs can lead to different behavior.[9] We assume the difference is the same order of magnitude as different random seeds would introduce.

With Jam-ADS we can see that larger equipment rates produce in general a higher average velocity and less fuel consumption. The only exception in the fuel consumption belongs to the outlier of the summary plot in Figure 7.22, though it should not be considered.

7.6.2. Number of lanes dependency

In Figure 7.24 a summary plot depending on the number of lanes is shown. In general the average velocity grows with the number of lanes with the exception of some runs of SUMO/Shawn with Jam-ADS whose average velocity is slightly smaller with three lanes than with two lanes.

The average fuel consumption has a minimum at two lanes for SUMO/Shawn without Jam-ADS[10] while CircSim without Jam-ADS has the same average fuel consumption for two and three lanes. With Jam-ADS SUMO/Shawn exhibits consumption increasing with the number of lanes. CircSim with Jam-ADS has a maximum at two lanes in a few runs, while other model-parameter sets have nearly the same consumption for all numbers of lanes.

Comparing the green with the blue lines (SUMO/Shawn with and without Jam-ADS) and the red with the black lines (CircSim with and without Jam-ADS) shows that Jam-ADS is beneficial for all tested numbers of lanes.

7.6.3. Jam-ADS distance dependency

Figure 7.25 contains a summary plot depending on the Jam-ADS distance. This is again a model-parameter which only has an influence if Jam-ADS is turned on. Contrary to the equipment rate dependency it exhibits no influence on SUMO/

[9] SUMO makes use of C++ associative containers (`std::map`, ...) with the IDs as keys. These containers store the keys in non-descending order based on comparison. The new unordered associative containers of C++11 [64] are not used in the SUMO versions applied in this work, however, even for those the iteration order may depend on the keys.

[10] Again, the only exception belongs to the outlier in Figure 7.22.

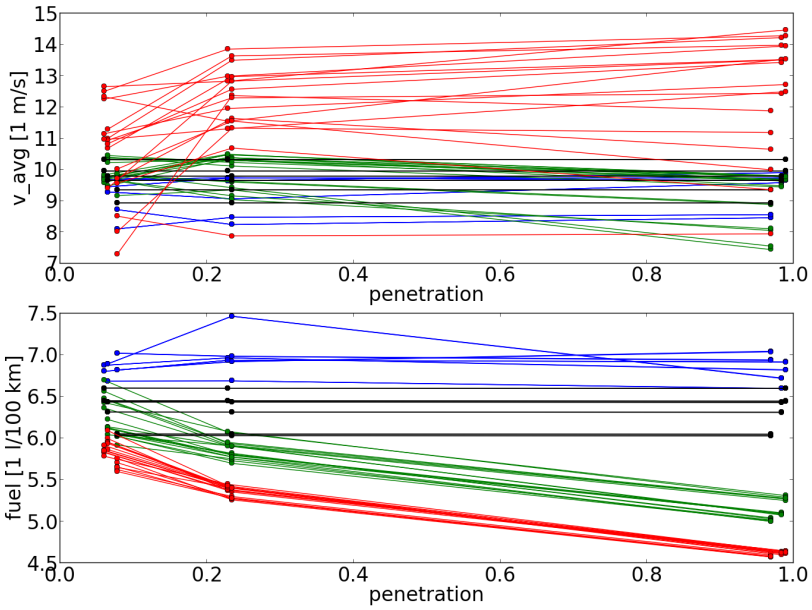


Figure 7.23.: Equipment-rate dependency of aggregated velocities and fuel consumptions.

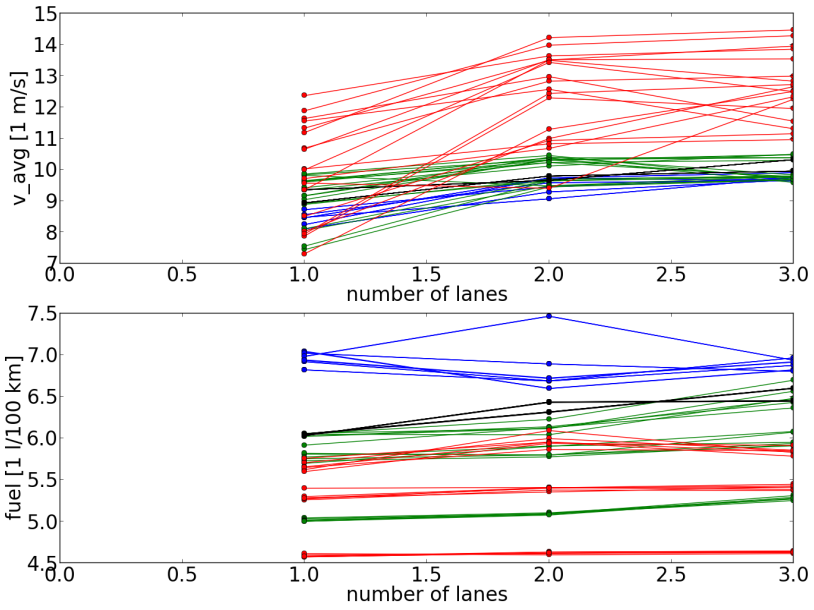


Figure 7.24.: Number-of-lanes dependency of aggregated velocities and fuel consumptions.

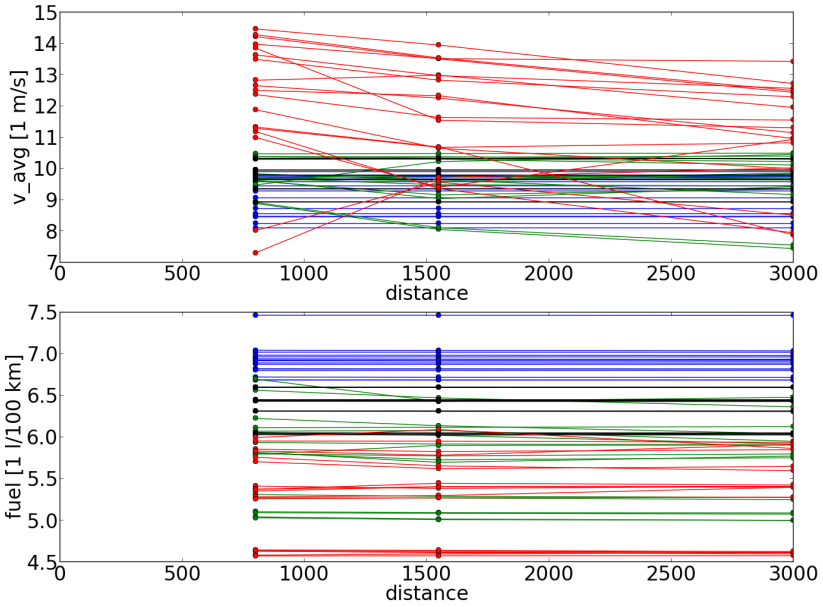


Figure 7.25.: Jam-ADS-distance dependency of aggregated velocities and fuel consumptions.

Shawn's behavior, thus all lines without Jam-ADS (blue and black) are exactly horizontal.

We can see that the average velocity has a tendency to decrease with increasing Jam-ADS distance, especially for CircSim (red lines). The exceptions belong to very low equipment rates of around 5 %. The Jam-ADS distance has nearly no influence on the average fuel consumption.

Together with the results of the equipment-rate dependency we see that for low equipment rates we should use a larger Jam-ADS distance and reduce it when the equipment rate increases. For a given mean vehicle density the product of Jam-ADS distance and equipment rate is proportional to the expectation value of the number of vehicles the Jam-ADS average is taken over. Further investigation is necessary to see whether there is a density depending optimal product of equipment rate and Jam-ADS distance.

8. The Open System

In the last chapter we showed that Jam-ADS works well in a closed system. Such systems are well investigated in academic traffic research, thus it was important to test the effectiveness of Jam-ADS in such a setting. However, closed systems are less realistic.

According to Helbing [51, p. 1112], the persistent spontaneous traffic jams of the closed system only appear because in a closed system the vehicle density can be set high enough to enter the region of linear instability. Conversely, in a system with open boundary conditions it is not possible to reach such a high density by increasing the inflow as long as there are no other inhomogeneities.

In addition, in a circular system inflow and outflow of a traffic jam are not independent of each other, but the relation of inflow and outflow is mainly determining the development of traffic jams, see Section 2.2.

Thus, to test Jam-ADS in a more realistic setting we executed many simulations with Jam-ADS in an open system as well. This chapter presents the main results of these simulations.

With the open system we executed 58 documented simulation-sessions taking more than 1076 hours of computation and whose data occupy 340 GiB of storage space. The simulated vehicles traveled more than 120 million km[1] and we rendered more than 35 000 plots.

8.1. Setup

Traffic jams are caused by some perturbation. A source for such could be obstacles, ascending slopes, on-ramps, off-ramps, etc. [127]. Obstacles can be for example accidents, road works, or the closing of lanes.

Ascending slopes broaden the distribution of velocities, because they become more dependent on the engine power in relation to the weight of the vehicle. As we have already seen, broader velocity distributions make congestion more likely. The opposite is the case for descending slopes.

Ramps are locations where traffic flows split or merge. While even off-ramps may cause traffic jams, on-ramps are more likely causes. Particularly, if the incoming flow on the highway and the flow on the on-ramp are both below their respective capacity, but the joined flow after the on-ramp is close to or above the capacity.

[1] We did not measure the traveled distance of the early open system simulation sessions.

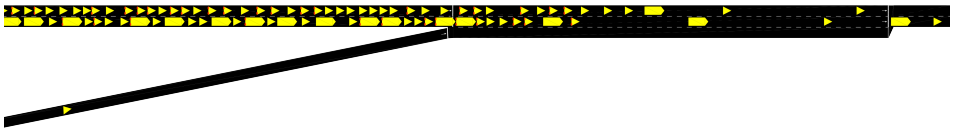


Figure 8.1.: Open system road network of our simulations. The picture exhibits a small cut-out of the complete road, showing the acceleration lane. The picture is generated with SUMO's visualization tool `sumo-gui`.

We selected an on-ramp as source of perturbation for our simulations of the open system. We are not interested in congestion caused by obstacles, because accidents are rare and road-works or closed lanes are known in advance and can be handled differently. Congestion caused by on-ramps has been well investigated and is easy to simulate with SUMO/Shawn. Ascending slopes are less often observed as congestion source and SUMO currently does not consider geographical elevation at all.

We do not apply CircSim for open system simulations, because it is not possible with reasonable effort to extend CircSim to simulate an open system. Implementing open-system simulations in CircSim would require a complete rewrite of the software, because the basic data structures of CircSim rely on a fixed number of vehicles (Appendix A.1).

Our road consists of a highway section with a single one-lane on-ramp (Figure 8.1). Additionally, we define an *evaluation section* directly upstream of the on-ramp. All spatial evaluation averages are taken over this section and the HDC introduced in Section 8.3 covers the same section. Table 8.1 contains the lengths of all sections.

As described in Section 6.2.1, SUMO inserts vehicles on the first edges of their routes. We want to achieve inflows as large as possible, thus we enable SUMO to insert vehicles as freely as possible on their *enter* edges and with arbitrary velocity. To separate insertion from the sections we want to observe, we add special *enter* edges upstream of the highway and the on-ramp. In addition, we need a *leave* edge, because SUMO removes vehicles from the road as soon as they enter the last edge of their route.

On the main highway we add several successive *enter* edges, because SUMO sometimes produces a collision by insertion and then teleports the offending vehicle to the start of the next edge of its route. Several *enter* edges avoid teleportation into the main highway edge that we evaluate for congestion. The number of *enter* edges can be configured in `run.cfg` (Section 6.4.2) and we found that three is sufficient for our experiments.

We tweaked the vehicle insertion options of SUMO to make the upper limit of possible inflows into the *enter* edges as large as possible. However, this does not

Table 8.1.: General configuration settings for the open system.

Setting	Value
Upstream section length	12 km
Acceleration lane length	200 m
Downstream section length	5 km
Number of highway lanes	2
Evaluation section length	8 km
Main flow	1500 vehicles/h/lane
Ramp flow	280 vehicles/h/lane
Maximum deceleration	4.5 m/s^2
Maximum velocity on highway	180 km/h
Maximum velocity entering on on-ramp	100 km/h
Passenger car length	5 m
Maximum acceleration of passenger cars	1.5 m/s^2
Truck length	12 m
Maximum acceleration of trucks	0.4 m/s^2

guarantee that SUMO is able to generate the inflow assigned to a simulation run. We measured the actual inflow of each simulation run and compared it to the assigned inflow, making sure to stay below the upper limit of SUMO.

We tested many different main and on-ramp flows and maximum velocities to discover which inflows produce which types of traffic jam. We present simulations with the flows and maximum velocities listed in Table 8.1, because they produce massive homogeneous congested traffic (HCT, see Section 2.2) and we show that Jam-ADS is effective even under that condition. These values are also in the order of empirically observed highway flows (for example, see Brilon [9]).

A problem we observed in SUMO/Shawn simulations is the missing synchronization of congestion between neighboring lanes. Some simulation runs exhibit a massive traffic jam on the right lane while vehicles on the left lane pass it with nearly maximum speed. To reduce this effect, we introduced new types of passenger cars and trucks with different maximum velocities. See Table 8.2 for our choice of the velocity distribution, while the remaining vehicle configuration is given in Table 8.1. This also causes a new distribution of both maximum acceleration and random deceleration at a given current vehicle speed. However, we have already seen—justified in Section 7.4—that Jam-ADS works on multiple lanes nearly as good as on a single lane.

Table 8.2.: Vehicle types of the simulation runs presented in this chapter with their fractions and maximum velocities.

Vehicle type	Fraction	Maximum velocity
car30	10 %	30 m/s = 108 km/h
car35	20 %	35 m/s = 126 km/h
car40	25 %	40 m/s = 144 km/h
car45	20 %	45 m/s = 162 km/h
car50	15 %	50 m/s = 180 km/h
truck80	8 %	22.2 m/s = 80 km/h
truck90	2 %	25 m/s = 90 km/h

8.2. Continuous Jam-ADS

For comparison with the closed system we first investigate Jam-ADS averaging over a fixed distance ahead of the vehicle in question, like we did for the closed system. To distinguish this from the other variants below, we call this *continuous Jam-ADS* (some plot labels contain the synonym *JamAdsClassic*).

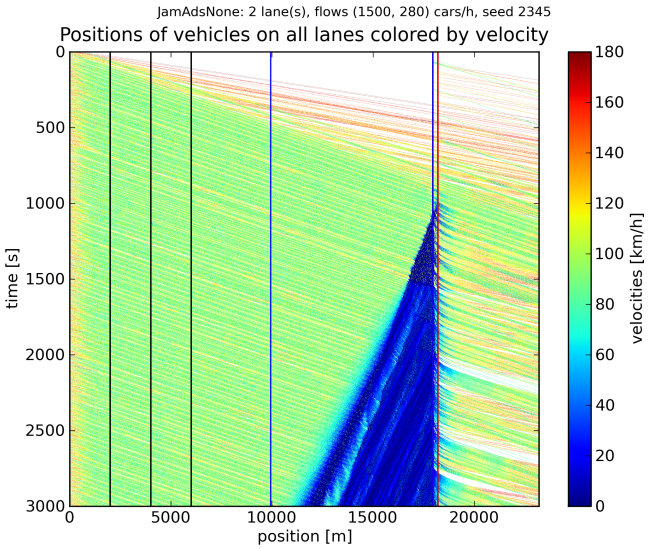
After some initial simulation experiments we found that we need to exclude the area of the on-ramp and further downstream, because the on-ramp section is already a flow bottleneck and Jam-ADS makes it worse by preventing vehicles that are merging into the highway from accelerating appropriately. Thus, in the following experiments the influence of Jam-ADS always ends at the downstream end of the on-ramp.

We now present some typical plots produced by simulations of continuous Jam-ADS in the open system. All presented simulation runs are executed with $\lambda = 0.67$ and a Jam-ADS distance of 8 km.

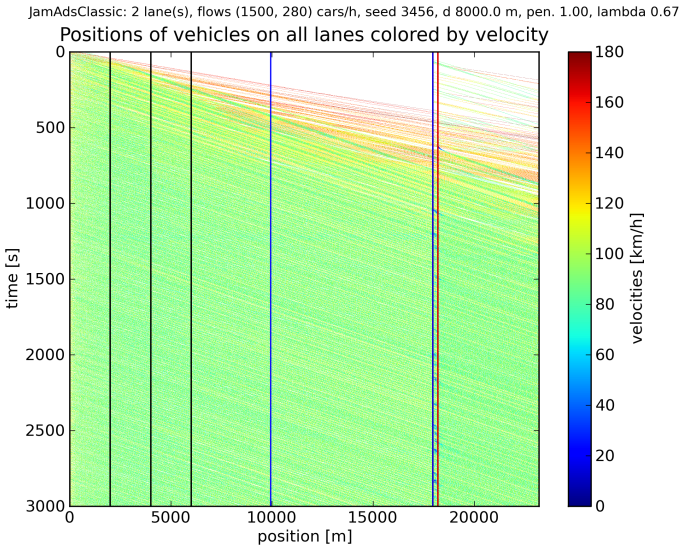
8.2.1. Full equipment

We begin with the positions plot to give an impression how Jam-ADS behaves within the open system (Figure 8.2). It shows the vehicles on the main highway lanes only, excluding vehicles on the on-ramp's *enter* edge and the acceleration lane.

The vertical lines mark the borders of SUMO's edges or other noteworthy locations. Between the left axis and the three black lines on the left are the three *enter* edges explained in Section 8.1. The two very close lines in blue and red around 18 000 m mark the beginning and the end of the 200 m long on-ramp's merge section. Thus, between the right black line at 6000 m and the right blue line at 18 000 m lies the 12 km long normal section upstream of the on-ramp. Between the red line and the right axis we have the 5 km long highway section downstream



(a) Without Jam-ADS.



(b) Continuous Jam-ADS at 100 % equipment rate.

Figure 8.2.: Positions plots without Jam-ADS and with continuous Jam-ADS at 100 % equipment rate.

of the on-ramp. Finally, between the blue lines at 10 000 m and 18 000 m is the 8 km long evaluation section over which we take velocity and fuel consumption averages. Note that the rightmost edge from which SUMO removes the vehicles is not shown, because SUMO removes the vehicles at the start of the final edge, hence it is empty.

The positions plot in Figure 8.2a shows the behavior without Jam-ADS. A short time after the system has settled a perturbation happens at the on-ramp which is large enough to trigger a growing traffic jam. After this starts to grow, visibly fewer vehicles leave the on-ramp section, because the outflow of a traffic jam is smaller than the maximum free flow (capacity) of a highway, see Section 2.2.

Figure 8.2b contains a positions plot of the same simulation executed with continuous Jam-ADS at 100 % equipment rate and $\lambda = 0.67$. The Jam-ADS distance of 8 km is cut off at the upstream end of the on-ramp's acceleration lane and Jam-ADS completely turned off at the on-ramp[2] and further downstream, because simulations revealed that Jam-ADS within the on-ramp and further downstream narrow down the outflow of the on-ramp, which makes dissolving of traffic jams less likely, as mentioned above. We see that no congestion occurs any more.

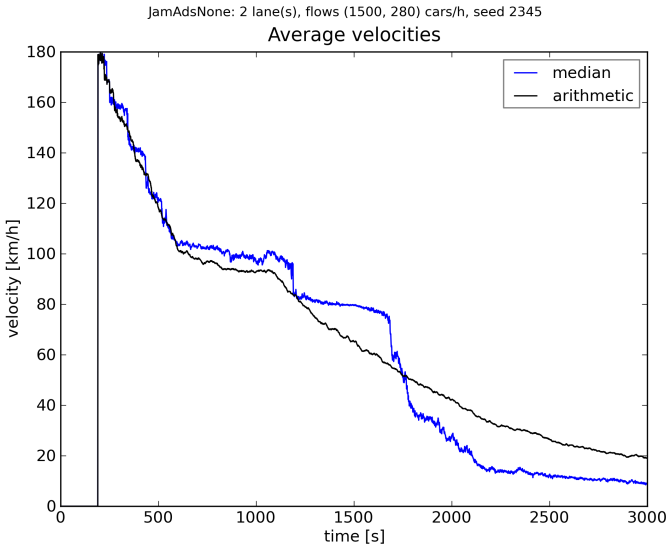
Figure 8.3 shows a plot of the development of the average velocity measured over the evaluation section. As usual the black line is the arithmetic average and the blue line the median. During the first approximately 200 seconds no vehicle reaches the evaluation section, as can be seen in the positions plots (Figure 8.2), hence there is no average during that time. The first vehicle entering the evaluation section is a passenger car with $v_{\max} = 180\,\mathrm{km/h}$. It has no vehicles ahead, thus the average jumps to 180 km/h at that moment, but soon decreases as more vehicles enter the evaluation section, particularly when the first trucks arrive there. At approximately 800 seconds the evaluation section is completely populated with vehicles.

The plot in Figure 8.3a shows that the average velocity monotonically decreases when the traffic jam starts growing at approximately 1400 seconds. Usually, the blue line of the median is nearly constant for some time after the jam begins to grow and falls abruptly to a very small velocity when more than half of the vehicles within the evaluation section are trapped in the traffic jam. In the presented plot there is an intermediate phase where coincidentally half of the vehicles are in the jam state for some longer time.

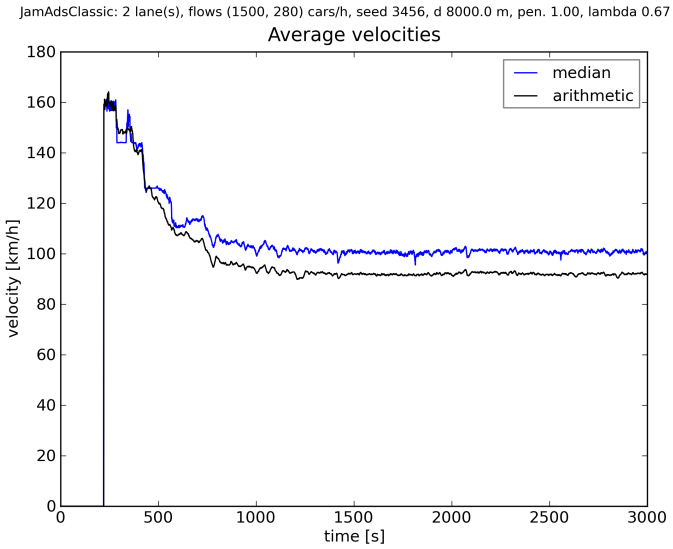
The average velocity plot in Figure 8.3b shows that continuous Jam-ADS keeps the average velocity constant once the stationary state has been reached. The median being above the arithmetic average means that more than half of the vehicles are faster than the average.

The plot in Figure 8.4 is a different view on the average velocity. It exhibits the average time it takes to travel through the entire evaluation section (see Sec-

[2] Technically the Jam-ADS distance is set to zero.



(a) Without Jam-ADS.



(b) Continuous Jam-ADS at 100 % equipment rate.

Figure 8.3.: Average velocity plots without Jam-ADS and with continuous Jam-ADS at 100 % equipment rate.

tion 6.4.2 for details) for passenger cars and trucks. Note that travel-time plots are retrospective not only by the moving average but also by considering at each point in time only the vehicles which completely passed the evaluation section.

We see in Figure 8.4a that the average travel time grows permanently because of the growing traffic jam. The intermediate peaks come from trucks which manage to enter the passing lane. Those trucks slow down the vehicles behind, thus they make all vehicles take longer to pass the evaluation section. The larger the traffic jam becomes, the smaller are the peaks, because within the traffic jam the trucks do not cause a slow-down of passenger cars, thus they have less influence.

Not that sometimes the travel time of the trucks drops to zero. This happens when no trucks pass the evaluation section for more than the moving-average length of 60 seconds.

With Jam-ADS the travel times are much shorter, as can be seen by the plot in Figure 8.4b. The stationary travel times are about the same as without Jam-ADS before the jam appears, which is around 1000 seconds in the plot in Figure 8.4a.

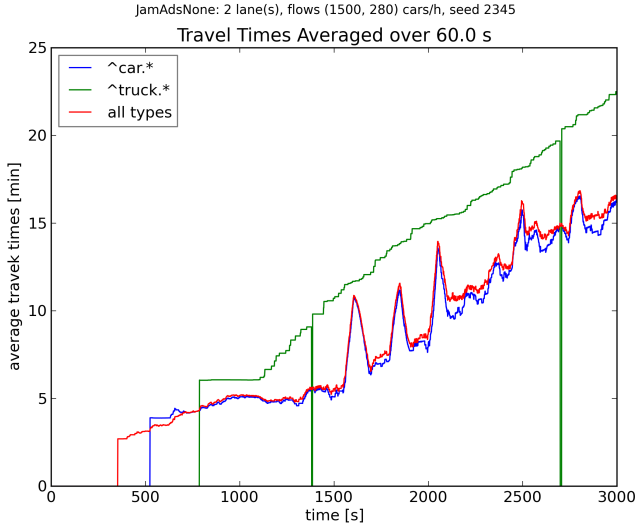
Figure 8.5 is a plot of the development of the mean fuel consumption within the evaluation section. Without Jam-ADS (Figure 8.5a) the fuel consumption is around 8 l/100 km. With Jam-ADS (Figure 8.5b) we have constant fuel consumption. With approximately 7 l/100 km it is about 1 l/100 km less than without Jam-ADS.

Before the traffic jam begins to grow at approximately 1500 seconds we have without Jam-ADS a fuel consumption of more than 9 l/100 km. Thus, Jam-ADS saves even fuel with respect to the consumption of free traffic without Jam-ADS.

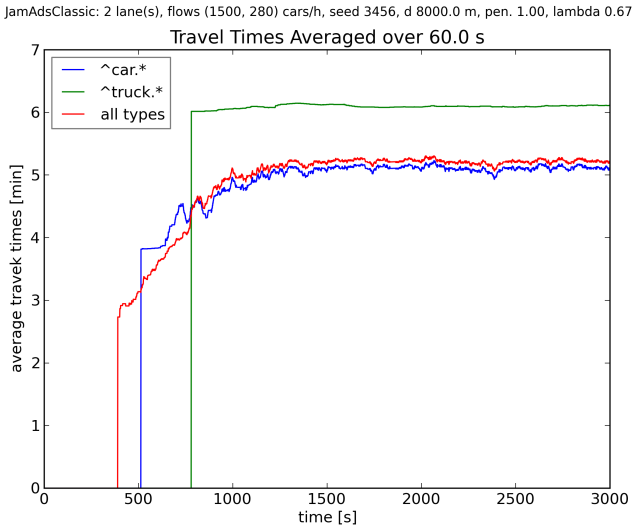
Figure 8.6 contains traffic flow plots of the system. Note that the assignment of colors to the different flows is not fixed. Instead, it depends on the arbitrary order in which the plot script discovers vehicles entering or leaving sections without coming from a preceding section or moving to a successive section. The plot in Figure 8.6a confirms that SUMO is able to perform the configured inflows. Both inflows (blue for highway, green for on-ramp) oscillate closely around the values given in Table 8.1. The outflow (red line) drops dramatically when the jam begins to grow at approximately 1500 seconds.

Figure 8.6b shows the same plot with Jam-ADS turned on. Here, the outflow is assigned a cyan color, because there is a tiny inflow directly into the section downstream of the on-ramp, where a teleportation occurred (Section 6.2.1), which received the red color.[3] With Jam-ADS the outflow is approximately the sum of both inflows, thus Jam-ADS rises the capacity (maximum flow) of the highway to meet the entire flow demand. However, Jam-ADS decreases the inflow slightly below the configured inflow.

[3] The colors are assigned automatically to all detected flows.

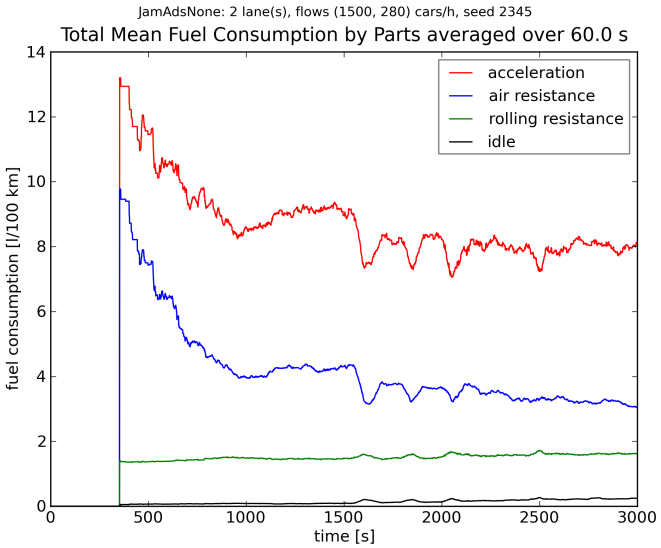


(a) Without Jam-ADS.

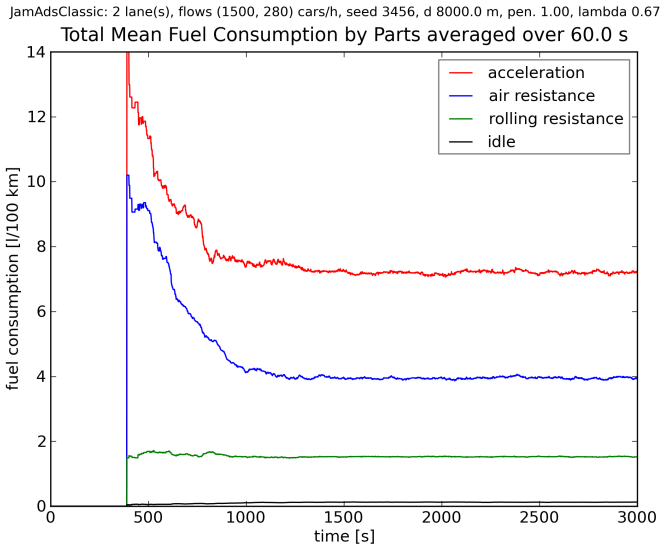


(b) Continuous Jam-ADS at 100 % equipment rate.

Figure 8.4.: Travel-times plots without Jam-ADS and with continuous Jam-ADS. The vehicle types of Table 8.2 are grouped by passenger cars (blue) and trucks (green) by applying the regular expressions shown in the legends. The red line gives the average over all vehicles, which is close to the passenger cars' average, because there are 88 % passenger cars.

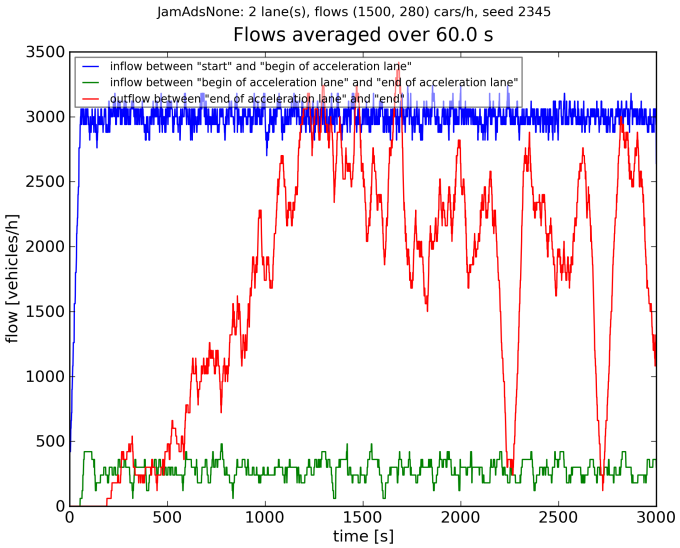


(a) Without Jam-ADS.

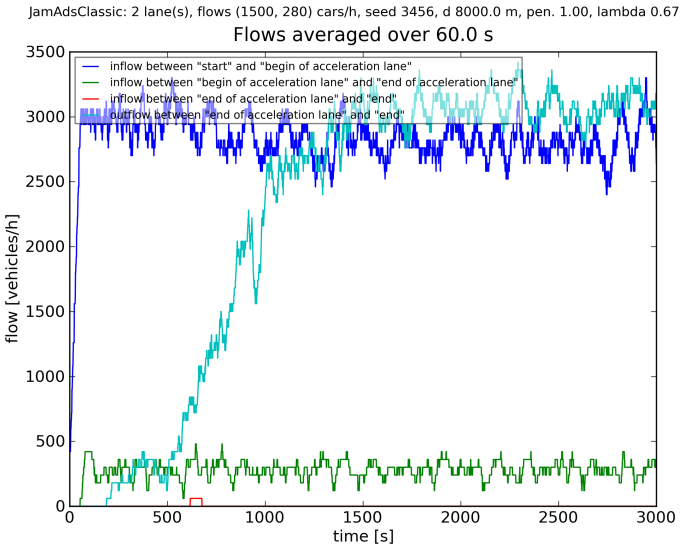


(b) Continuous Jam-ADS at 100 % equipment rate.

Figure 8.5.: Average fuel-consumption plots without Jam-ADS and with continuous Jam-ADS at 100 % equipment rate.



(a) Without Jam-ADS.



(b) Continuous Jam-ADS at 100 % equipment rate.

Figure 8.6.: Vehicle-flow plots without Jam-ADS and with continuous Jam-ADS at 100 % equipment rate.

8.2.2. Limited equipment rate

We now investigate continuous Jam-ADS with a limited equipment rate of 5 % and 30 %. Again, we begin with the positions plots (Figure 8.7) to give a general impression.

At 30 % equipment rate (Figure 8.7a) the strategy works most of the time like 100 % equipment rate, but sometimes vehicles need to slow down at the on-ramp, meaning that upcoming equipped vehicles have to slow down as well. This deceleration propagates very quickly upstream causing a kind of distributed traffic jam. After a few minutes however, this jam is dissolved. Thus, on average, the situation is better than without Jam-ADS.

At 5 % equipment rate (Figure 8.7b) the effects of Jam-ADS look differently. A congestion in the on-ramp section also propagates quickly to all equipped vehicles upstream making them move slower, but too many unequipped vehicles enter into the emerging large gaps. The traffic jam, which without Jam-ADS was located upstream of the on-ramp (Figure 8.2a), is now distributed over the whole upstream area and accumulates within the highway enter edges in which SUMO inserts more and more vehicles into the gaps.

Figure 8.8 contains a plot of the development of the average velocities for the limited equipment rate within the evaluation section. At 30 % (Figure 8.8a) the free traffic velocities are kept a bit longer until the special jams start, as we have seen in the positions plot. Each of them causes a quick drop of the average velocity with a following, nearly complete, recovery.

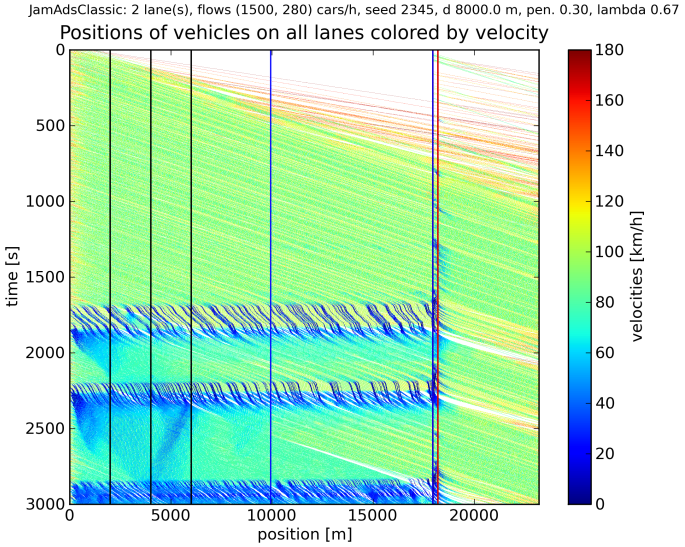
The median shows that at the start of each drop less than half of the vehicles slow down and then, in a later step, the others follow. Like at the other experiments, the leveling effect of Jam-ADS makes the median follow the arithmetic average closely.

At 5 % equipment rate (Figure 8.8b) there is no complete recovery after the velocity drops for the first time after the initial settling. Instead, the average velocity oscillates around approximately half of the free traffic average velocity. Thus, on average, the vehicles are—even at this low equipment rate—faster than without Jam-ADS.

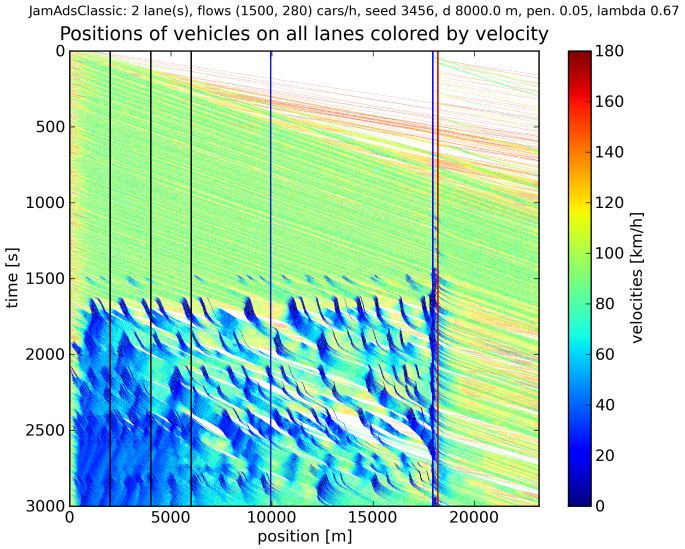
The travel times (Figure 8.9) are worse than with full equipment but considerably better than without Jam-ADS. Even with only 5 % equipment rate the travel time is bounded, but with oscillations. At 30 % there are less oscillations.

The plot in Figure 8.10 exhibits the mean fuel consumption within the evaluation section at limited equipment rate. At 30 % (Figure 8.10a) we have slightly less consumption than without Jam-ADS and at 5 % (Figure 8.10b) there are no more fuel savings because of Jam-ADS.

Finally, in Figure 8.11, we present the flows plot at limited equipment rate. The inflow at 30 % equipment rate (Figure 8.11a) exhibits an interesting behavior. During the jam phases the total inflow first drops but then rises for a short time

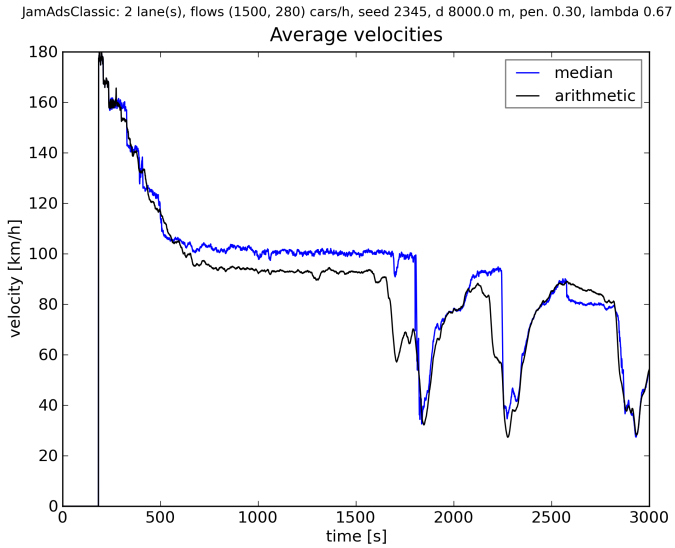


(a) 30 % equipment rate

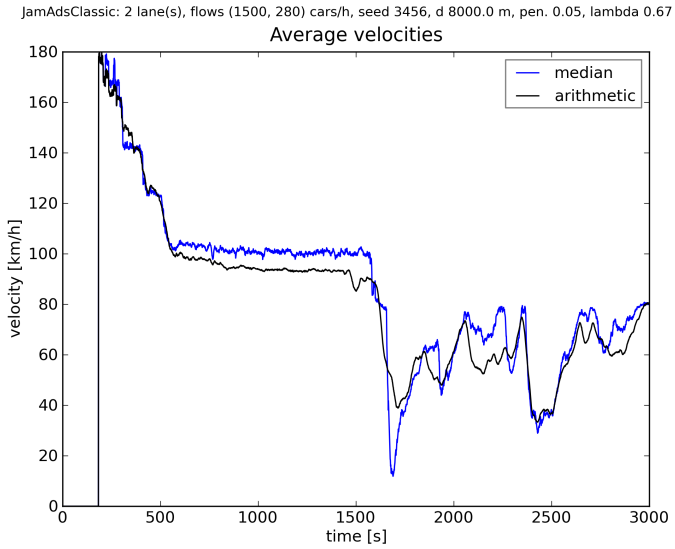


(b) 5 % equipment rate

Figure 8.7.: Positions plots of continuous Jam-ADS at limited equipment rates.

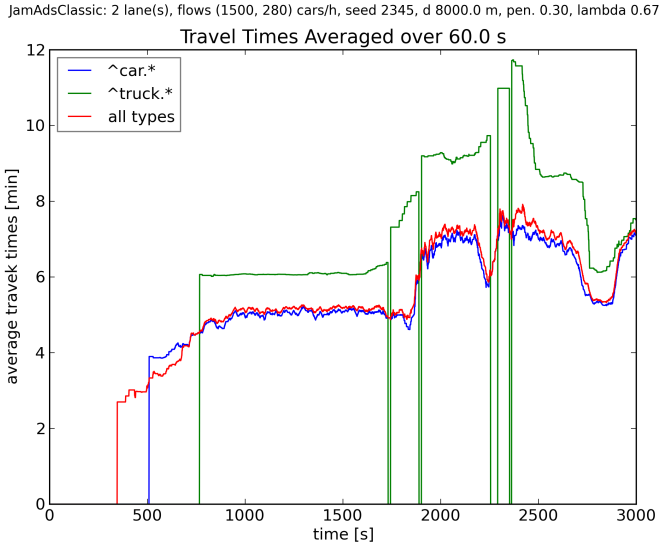


(a) 30 % equipment rate

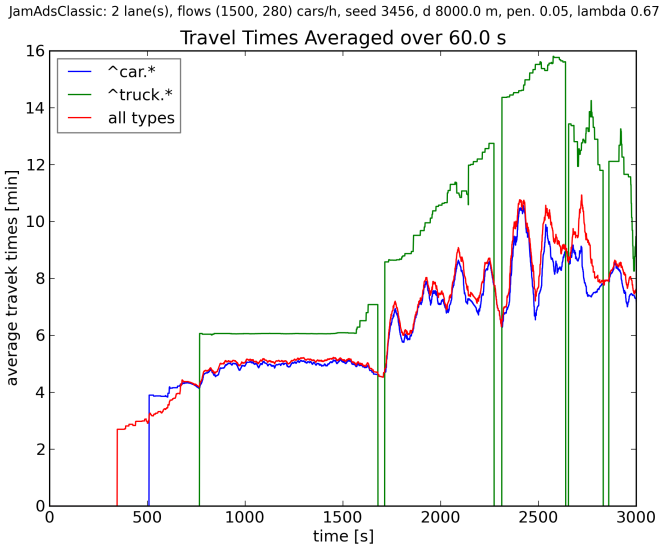


(b) 5 % equipment rate

Figure 8.8.: Average velocities within the evaluation section of continuous JamADS at limited equipment rates.

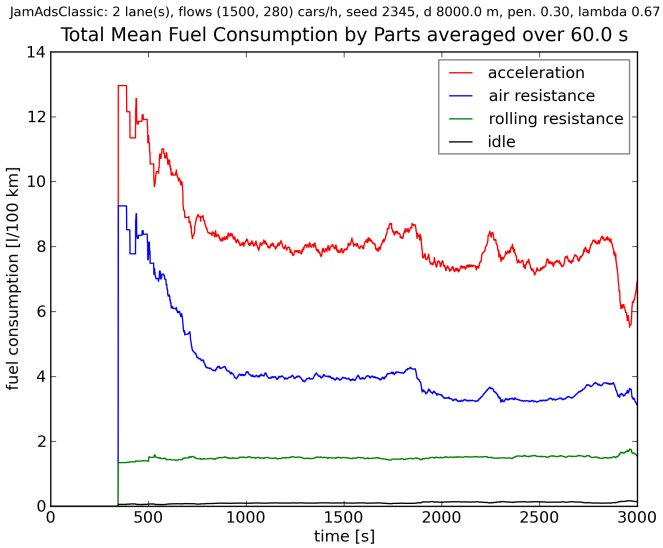


(a) 30 % equipment rate

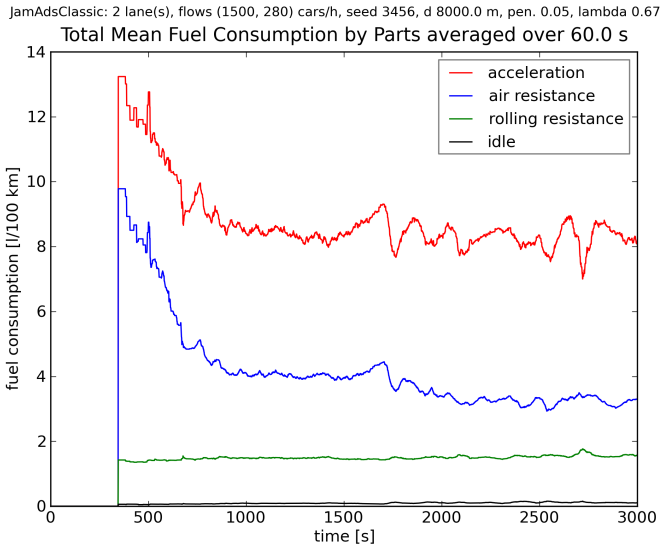


(b) 5 % equipment rate

Figure 8.9.: Travel-times plots with Jam-ADS at limited equipment rates.



(a) 30 % equipment rate



(b) 5 % equipment rate

Figure 8.10.: Mean fuel consumptions within the evaluation section of continuous Jam-ADS at limited equipment rates.

much higher than the configured inflow of 3000 vehicles/h. The reason for this is as follows: The deceleration caused by the congestion over the entire upstream section first prevents SUMO from inserting the planned number of vehicles, which causes a backlog of vehicles waiting for insertion (see Section 6.2.1). When soon after the gaps in front of the slow equipped vehicles emerge, SUMO manages to insert all the waiting vehicles causing a high inflow. The outflow exhibits drops during the congestion phases, but recovers as well, such that all vehicles of the highway and the on-ramp pass through.

At 5 % equipment rate (Figure 8.11b) complete recovery does not occur any more and we see some oscillations. Especially the inflow into the highway is reduced in total, but in reality we do not have an influence on this boundary condition. Thus, we have to presume a less positive effect of Jam-ADS with 5 % equipment rate if the inflow stays at the configured 3000 vehicles/h.

8.3. HDCs

As next step towards the HDC concept (Section 5.1) we limit the area where Jam-ADS is active to a special section we call *HDC-area*.[4] In our open system setting we selected the section 8 km upstream of the on-ramp as HDC-area, which we already used as evaluation section for continuous Jam-ADS (the section between the blue lines in the positions plots) as interesting congestion phenomena occur within this section. In addition, it makes comparison with the continuous Jam-ADS results easier.

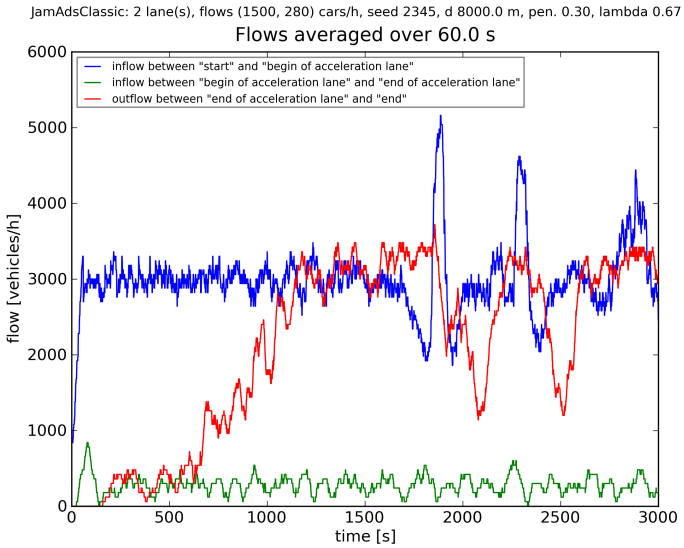
In reality, we assume that a Jam-ADS device has access to the road maps of the navigation device, thus Jam-ADS knows about oncoming ramps and other sources of congestion, such that it can set up an appropriate HDC at congestion-critical road sections.

In our first tests with this limit, we took the Jam-ADS average always over the whole HDC-area regardless of the considered vehicle. This turned out to be counterproductive, because slow equipped vehicles upstream made equipped vehicles further downstream decelerate, causing more congestion than without Jam-ADS. Thus, even with the limitation to the HDC-area we always take the Jam-ADS average over vehicles ahead.

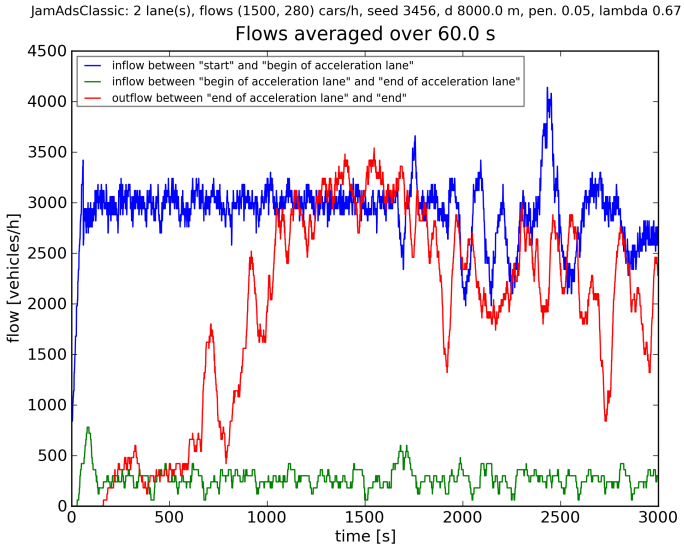
8.3.1. Variants

The limitation of Jam-ADS to the HDC-area revealed some problems, which we met with several variants of the strategy. First, we distinguish between *observed*

[4] The next step towards the HDC/OIC concept would be to make the HDC-area arise from developing congestion (see outlook in Section 11.2).



(a) 30 % equipment rate



(b) 5 % equipment rate

Figure 8.11.: Flows of continuous Jam-ADS at limited equipment rates.

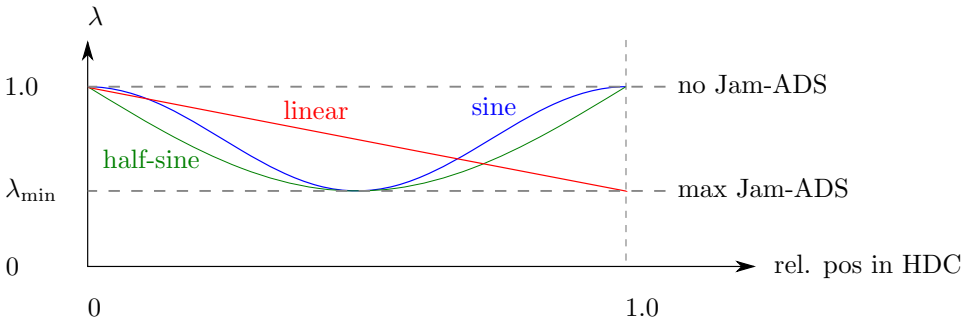


Figure 8.12.: Different dependencies of λ on relative position within HDC-area.

and *controlled areas* of the HDC.[5] The HDC collects its data from vehicles within the observed area and applies it to vehicles within the controlled area. The controlled area remains to the mentioned 8 km upstream of the on-ramp, but for some simulation runs we extended the observed area downstream to cover the on-ramp section, because jams are triggered by perturbations within the on-ramp section.

Next, we observed in some experiments that Jam-ADS shifted traffic jams to the upstream border of the HDC-area. The plot in Figure 8.16a gives an idea how this looks like in a positions plot, although the plot belongs to a completely different strategy discussed below (Section 8.4). A possible reason might be that Jam-ADS starts abruptly at the upstream border of the HDC-area making equipped vehicles harshly decelerate when they enter the HDC-area and there is congestion ahead.

As a solution we make λ depending on the *relative position within the controlled HDC-area*, x_{rel}. We define x_{rel} to be 0 at the upstream border and linearly increasing to 1 at the downstream border of the HDC-area. At the upstream border of the controlled area, where we have $x_{\text{rel}} = 0$, we make sure that $\lambda(x_{\text{rel}} = 0) = 1.0$, which means no Jam-ADS influence according to Equation (5.1). The value of λ set in the simulation configuration acts now as minimum value λ_{min} within the controlled HDC-area to parametrize the maximum effectiveness of Jam-ADS.

As first variant of such a position dependent λ we make $\lambda(x_{\text{rel}})$ drop linearly from $\lambda(x_{\text{rel}} = 0) = 1.0$ to $\lambda(x_{\text{rel}} = 1) = \lambda_{\text{min}}$ within the controlled area as shown as the red line in Figure 8.12. This did not improve anything, because another problem are single equipped vehicles at the downstream end of the observed area, which have to brake heavily for arbitrary reason, thus making all equipped vehicles upstream of them within the controlled area brake as well.

To solve this problem, we make $\lambda(x_{\text{rel}})$ grow back to $\lambda(x_{\text{rel}} = 1) = 1.0$ at the downstream end of the controlled HDC-area. To do this, we implemented two

[5] This goes back to a suggestion by Sebastian Ebers.

more dependencies of the relative position denoted in Figure 8.12 as "sine" (blue line) and "half-sine" (green line). "sine" is a complete period of a sine curve

$$\lambda_{\text{sine}}(x_{\text{rel}}) = \lambda_{\min} + \frac{1}{2}\left(1 + \cos(2\pi x_{\text{rel}})\right)\left(1 + \lambda_{\min}\right) \quad , \tag{8.1}$$

whereas "half-sine" is half a sine period

$$\lambda_{\text{half-sine}}(x_{\text{rel}}) = 1 - (1 - \lambda_{\min})\sin(\pi x_{\text{rel}}) \quad . \tag{8.2}$$

The main difference between "sine" and "half-sine" for our purposes is the gradient at the HDC borders. While "sine" starts with a slow drop and ends with a slow raise "half-sine" makes λ drop more quickly and keeps it reasonably below 1 until nearly the end of the controlled HDC-area.

We did not implement linear dependency drop to $\lambda_{\min}$ at the center of the controlled area followed by a linear grow back to 1 at the end, because we wanted to have a broader area where λ takes values close to $\lambda_{\min}$.

In addition to the position dependency, we implemented a dependency of λ of the density of the vehicles ahead. Because Jam-ADS only knows about equipped vehicles, the implementation guesses the total density by the density of equipped vehicles and their fraction of all vehicles (i.e., equipment rate). We assume that in reality the equipment rate changes slow enough, such that the system knows it from other sources. For ACC equipped vehicles (see Section 4.6) it is possible to derive the equipment rate by observing the fraction of equipped predecessors.

If we denote the dependency of the relative position as $\lambda_x(x_{\text{rel}})$ (i.e., $\lambda_x \equiv \lambda_{\text{sine}}$ or $\lambda_x \equiv \lambda_{\text{half-sine}}$) and the dependency of the density as $\lambda_\rho(\rho)$ then we set $\lambda(x_{\text{rel}}, \rho)$ to

$$\lambda(x_{\text{rel}}, \rho) = 1 - \lambda_\rho(\rho)(1 - \lambda_x(x_{\text{rel}})) \quad . \tag{8.3}$$

This means, with $\lambda_\rho(\rho) = 0$ we turn off Jam-ADS and a growing $\lambda_\rho(\rho)$ increases Jam-ADS until we reach the previous $\lambda(x_{\text{rel}})$ without density dependency at $\lambda_\rho(\rho) = 1$. Figure 8.13 illustrates this for $\lambda_x \equiv \lambda_{\text{half-sine}}$.

We implemented two different λ_ρ functions. The first,

$$\lambda_{\rho,\ \text{const}}(\rho) \equiv 1 \tag{8.4}$$

is the behavior without density dependency for comparison to previous experiments. The second

$$\lambda_{\rho,\ \text{relative}}(\rho) = \frac{\rho}{\rho_{\max}} \tag{8.5}$$

sets λ_ρ in relation to a constant high density $\rho_{\max}$. Depending on the selection of $\rho_{\max}$ this function cannot warrant $\lambda_\rho(\rho) \leq 1$, because reasonable settings of $\rho_{\max}$ are smaller than the bumper-to-bumper density and the estimation of ρ for non-full equipment rate as described above brings further uncertainty of the density.

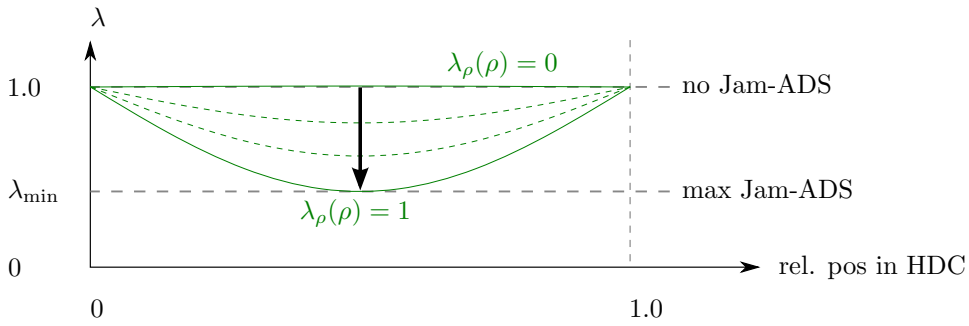


Figure 8.13.: Density dependency of $\lambda(x_{\text{rel}}, \rho)$ for $\lambda_x \equiv \lambda_{\text{half-sine}}$ and different values of $\lambda_\rho(\rho)$. The arrow points from $\lambda_\rho(\rho) = 0$ to $\lambda_\rho(\rho) = 1$.

However, we want the final $\lambda(x_{\text{rel}}, \rho)$ to obey $\lambda_{\min}$ as absolute minimum. To ensure this, $\lambda_\rho(\rho)$ is cut such that we always have $\lambda_\rho(\rho) \in [0, 1]$.

We mentioned above the effect of "averaging" over a single equipped vehicle ahead which braked hard for an arbitrary reason. Increasing λ back to one at the downstream end of the HDC-area did not entirely avoid this. So we tested another variant of Jam-ADS by turning it off when either the number or the density of the equipped vehicles over which the Jam-ADS average is taken is below a threshold, which is configurable as model-parameter.

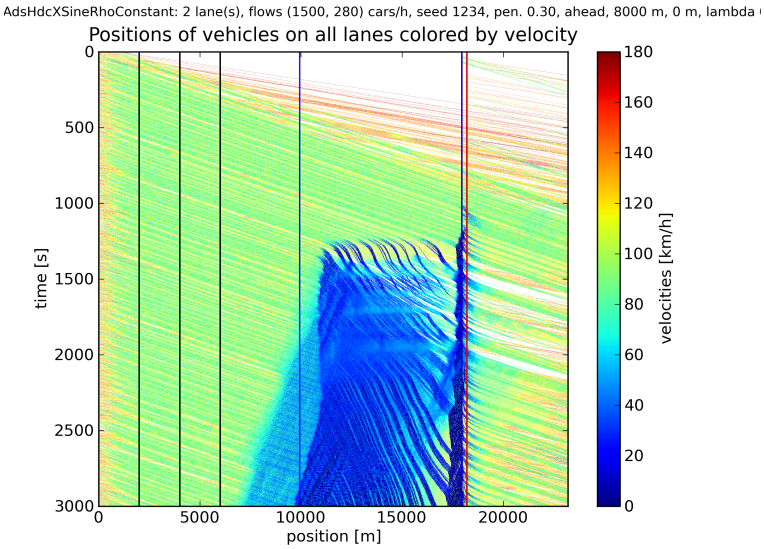
8.3.2. Some results

We tested the effects of all variants described in the last section in many different combinations with different parameter settings under different traffic demands. However, we do not present each of them here, because the differences are often negligible. Instead, we explain a few examples exhibiting the main results.

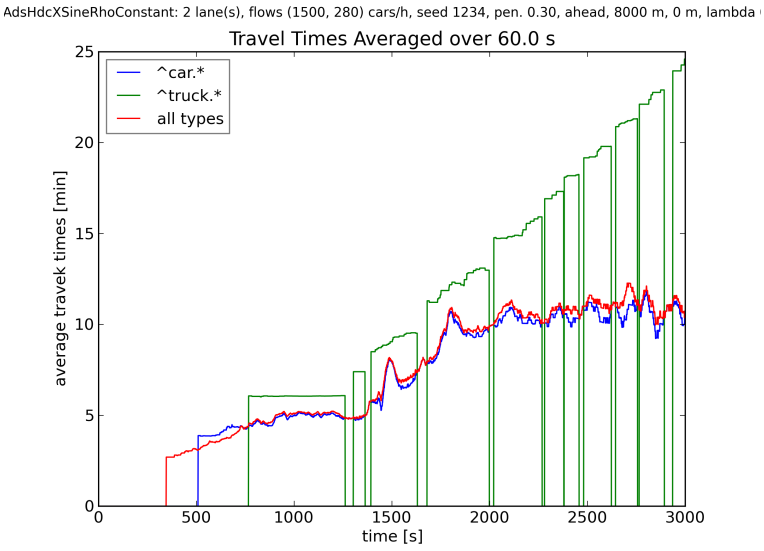
In Figures 8.14 and 8.15 we present typical results of a simulation run with Jam-ADS limited to an HDC-area. In this run, observed and controlled area are identical and set to the usual 8 km upstream of the on-ramp. The position dependency of λ is set to λ_{sine} with $\lambda_{\min} = 0.67$ (Equation (8.1)) and no density dependency (Equation (8.4)). The run is executed with 30 % equipment rate.

Figure 8.14a contains the positions plot. The traffic jam, which usually grows upstream of the on-ramp (see Figure 8.2a), is immediately distributed over nearly the whole HDC-area, but it is less dense than without Jam-ADS. A narrow dense jam grows and shrinks directly upstream of the on-ramp.

Figure 8.14b exhibits the plot of the mean travel-time development. The passenger cars reach a stationary travel time that is higher than the free traffic travel



(a) Positions plot



(b) Travel times

Figure 8.14.: Positions and travel times of Jam-ADS limited to HDC-area without further improvements.

time, while the travel time of the trucks continuously raises. As before, the travel time of the trucks drops to zero if for at least 60 seconds no truck finishes passing the evaluation section. A reason for the different behavior of passenger cars and trucks is that congestions produced by SUMO are different on the right and left lane and trucks mainly stay on the right lane. However, travel times with Jam-ADS limited to an HDC-area are worse than with continuous Jam-ADS.

The mean fuel consumption (Figure 8.15a) is up to 1 l/100 km smaller than without Jam-ADS (Figure 8.5a), although it tends to grow again.

Finally, the flows plot in Figure 8.15b shows that the inflows on both the main highway and the on-ramp are at the configured values of Table 8.1. At the end, the outflow of the highway is smaller than the inflow, but better than the outflow of the system without Jam-ADS (Figure 8.6a). Contrary to continuous Jam-ADS the inflow remains at the configured value.

Comparing simulation runs with all combinations of variants and different parameter settings, we came to the conclusion that the only relevant variant is the relative position dependency of λ in the forms "sine" or "half-sine". But there is no general advantage of one over the other. Also simulations with only 5 % equipment rate were not considerably worse than the presented 30 % equipment rate.

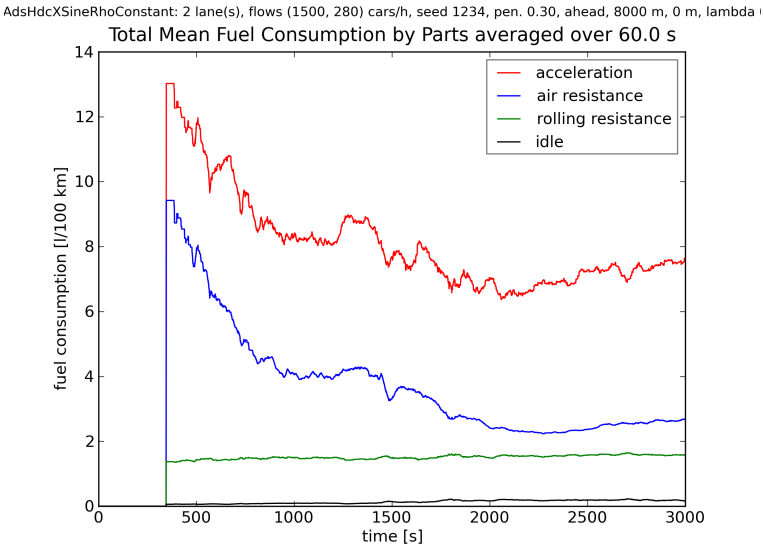
All other variants (particularly the density dependency) did not lead to a recognizable and reproducible improvement. Non-systematic differences between the simulation runs were the time, when the first congestion starts, the development of dense narrow jams upstream of the on-ramp, and if congestion develops upstream of the HDC-area.

8.4. Relax-strategy as alternative to average-strategy

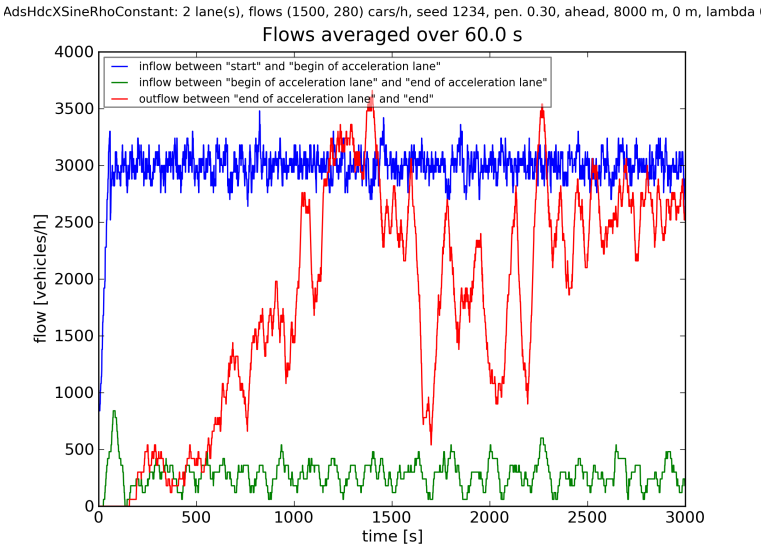
We have seen that in the tested variants the effects of Jam-ADS with the presented algorithm are not as good as expected if limited to an HDC-area only. To address this problem we developed and tested a completely different Jam-ADS algorithm that we present in this section. To distinguish both, we call the former algorithm *average-strategy* and the new one *relax-strategy*.

The experiments with average-strategy limited to the HDC-area also showed that recommending a velocity much smaller than the driver's desired speed causes new perturbations at the upstream end of the HDC-area, which often lead to congestion at that location. To overcome this, the new strategy does not recommend an absolute velocity. Instead, it only recommends to slightly decelerate by releasing the accelerator pedal more or less. This is why we call the strategy "relax".

An additional advantage of this slight deceleration are fewer safety problems as the average-strategy would have as a fully automatic application. Every driver on a highway has always to be prepared for a slight deceleration of the vehicle in front,



(a) Fuel consumption



(b) Flows

Figure 8.15.: Fuel consumption and flows of Jam-ADS limited to HDC-area without further improvements.

caused by releasing the accelerator pedal. Thus, it would be easier to implement the relax-strategy as a fully automatic system. The system may inform the driver when it comes into action and may give an override option.

In our simulation the relaxing is modeled by a *total relax-factor* r_{tot} between 0 and 1, which we multiply with the absolute maximum acceleration a of the considered vehicle, because we assume that the absolute of the maximum acceleration approximates the deceleration caused by completely releasing the accelerator pedal. This corresponds to the modeling of the random deceleration of the Krauß model (Equation (3.11)). The resulting relax deceleration is compared to the current random deceleration and the larger of both is applied to the vehicle. This is more realistic than applying both independently, because both are a result of releasing the accelerator—either intentionally or unintentionally—and have a joined limit. Particularly, a complete release of the accelerator prevents all random deceleration caused by slight changes of the accelerator pedal's position. The velocity recommended by the relax-strategy is defined as

$$v_{\text{rec}} = \min\left[v_{\text{des}} - r_{\text{tot}}\, a\Delta t,\ \text{rand}(v_{\text{des}} - \varepsilon a\Delta t,\ v_{\text{des}})\right] \quad , \tag{8.6}$$

which already includes the random deceleration (Equation (3.11)) of the Krauß model.

The recommended velocity v_{rec} is used as maximum velocity of the Krauß model in the next simulation step to continue relax deceleration over several steps. Otherwise, we would have only a slightly smaller velocity by relaxing. This continues until relaxing is turned off.

The total relax-factor is a product of a *general relax-factor* r_{gen} to configure the effectiveness of the strategy and a *special relax-factor* r_{spec} depending on the behavior of other vehicles defined by the concrete relax-strategy

$$r_{\text{tot}} = r_{\text{gen}}\, r_{\text{spec}} \quad . \tag{8.7}$$

We tested two special relax-strategies, which we present in the following sections.

As the relax-strategy recommends a deceleration instead of a certain velocity, a short but heavy slow down ahead would not cause a heavy slow down upstream as it is the case with average-strategy.

8.4.1. Relax-strategy depending on braking vehicles

The first specialization of the relax-strategy applies relaxing when a vehicle ahead within the observed area of the HDC actively brakes. A vehicle is considered braking if its absolute value of deceleration during the last time step is larger than its maximum acceleration a.

In this strategy the special relax-factor r_{spec} is defined as the relative position of our vehicle between the upstream end of the observed area of the HDC and the

closest braking vehicle ahead:

$$r_{\text{spec}} = \frac{x_{\text{my vehicle}} - x_{\text{upstream end of HDC}}}{x_{\text{braking vehicle}} - x_{\text{upstream end of HDC}}} \quad . \tag{8.8}$$

This means the closer our vehicle is to the closest braking vehicle ahead within the observed HDC-area the larger is the relax deceleration. This factor also makes sure that the strategy sets in smoothly when the vehicle enters the observed HDC-area.

Figure 8.16 contains plots of a simulation run with the usual configuration given in Table 8.1 and identical observed and controlled HDC-areas (not extending into the on-ramp), but with an added general relax-factor $r_{\text{gen}} = 1$. Jam-ADS shifts the traffic jam from directly upstream of the on-ramp (Figure 8.2a) to upstream of the controlled HDC-area, leaving nearly no congestion within the controlled area (Figure 8.16a).

The travel times (Figure 8.16b) are only a little longer than in free traffic, because the plot only considers the evaluation section, which is free of congestion most of the time. The little increase of the travel times comes from the fact that the vehicles enter the evaluation section at the very small velocity of the jam upstream.

The mean fuel consumption within the evaluation section (Figure 8.16c) is also a little higher than with continuous Jam-ADS, because of the acceleration of the vehicles when entering the evaluation section.

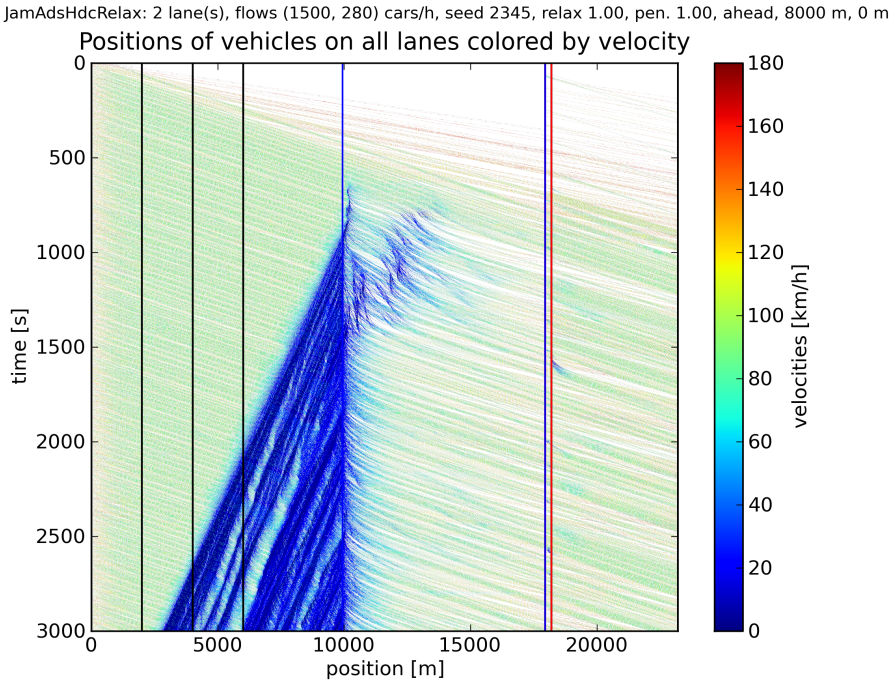
Travel times and fuel consumption would be much worse, if we would have measured them in the congested section upstream of the evaluation section, thus this configuration is disadvantageous.

Figure 8.17 contains plots of the same simulation with a general relax-factor $r_{\text{gen}} = 1/4$. In this case the traffic jam is nearly completely cleared away except for a short slow-down in vicinity of the upstream end of the controlled area (Figure 8.17a). The travel times (Figure 8.17b) are similar to those of continuous Jam-ADS, except for a slight increase because of the short slow-down. The same applies to the mean fuel consumption (Figure 8.17c).

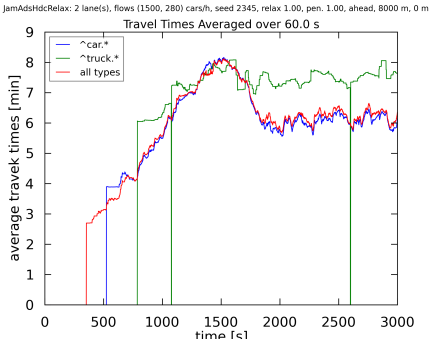
The simulation run presented in Figure 8.17 is a run in which the strategy behaved very well. In other runs, one or a few small jam waves are released upstream of the controlled area and move further upstream.

Further tests with different general relax-factors r_{gen} revealed, that general relax-factors below $r_{\text{gen}} = 3/4$ produce similar results. A general relax-factor $r_{\text{gen}} = 0.9$ works at least partially. The general relax-factor r_{gen} is the upper limit of the total relax-factor r_{tot} and a smaller total relax-factor r_{tot} means less deceleration due to the relax-strategy. Thus, the relax-strategy is more effective if its technical influence is smaller.

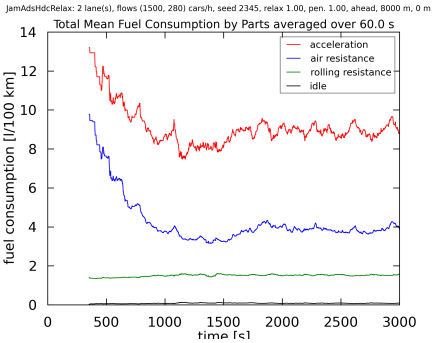
As last example of the braking relax-strategy we present what happens if the observed HDC-area is extended into the on-ramp section (Figure 8.18). Figure 8.18a



(a) Positions plot

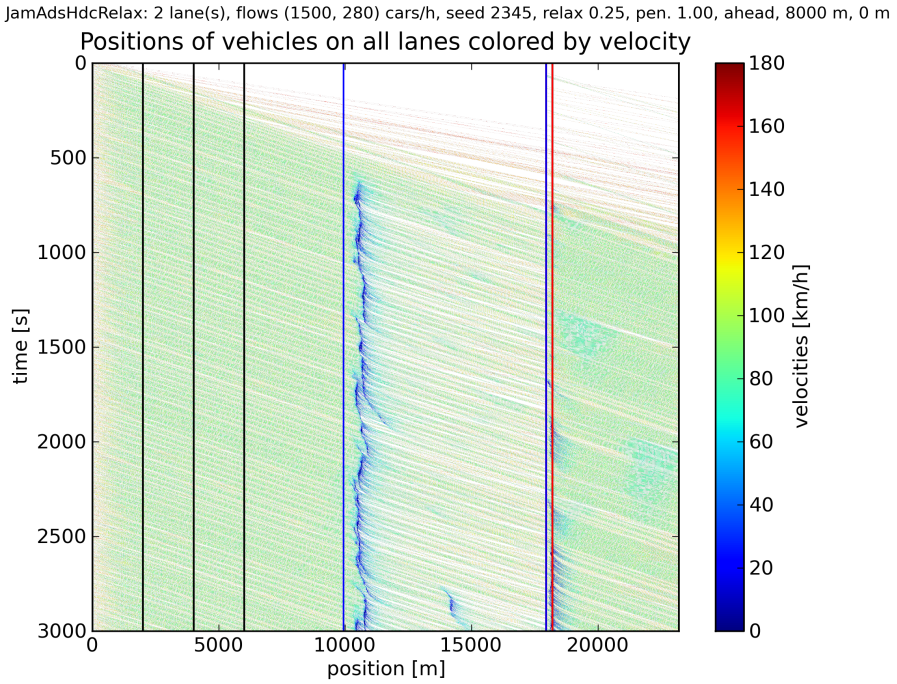


(b) Travel times in evaluation section (between the blue vertical lines above)

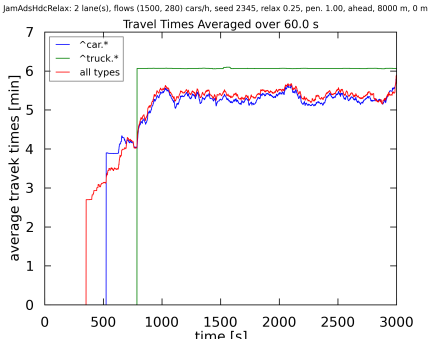


(c) Mean fuel consumptions in evaluation section (between the blue vertical lines above)

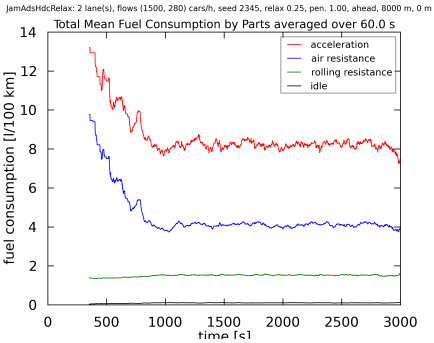
Figure 8.16.: Relax-strategy braking with a general relax-factor $r_{\text{gen}} = 1$.



(a) Positions plot



(b) Travel times in evaluation section (between the blue vertical lines above)



(c) Mean fuel consumptions in evaluation section (between the blue vertical lines above)

Figure 8.17.: Relax-strategy braking with a general relax-factor $r_{\text{gen}} = 1/4$.

shows massive slow down over the whole controlled area and a narrow dense traffic jam directly upstream of the on-ramp.

The travel-times plot in Figure 8.18b confirms the congestion by increased travel times. Because of the jam, the travel times of passenger cars and trucks are about the same and both end at twice the travel times of free traffic. The mean fuel consumption (Figure 8.18c) is much smaller than in free traffic, because the vehicles take on average more than 10 minutes to pass the 8 km long evaluation section, thus they are often too slow to drive in the highest gear. Our fuel consumption model does not consider the increased consumption of driving in lower gears [142]. However, we can see the oscillations caused by the frequent decelerations and accelerations. We conclude that for the braking relax-strategy an extension of the observed HDC-area into the on-ramp is counterproductive.

8.4.2. Relax-strategy depending on time headway

According to Helbing [51], the distribution of time headways is mostly independent of the velocity and shows a strong maximum at approximately one second. Moreover, Neubert et al. [109] suspect that small time headways indicate a transition to congested traffic. This led to the idea to use the average time headway of the vehicles ahead as main input parameter of another dependency of the special relax-factor r_{spec}.

The time headway relax-strategy takes the average time headway $t_{\text{avg headway}}$ of equipped vehicles ahead within the observed HDC-area as input parameter. If this is below a configured threshold t_{thresh}, the relation to the threshold is taken to compute the special relax-factor r_{spec}:

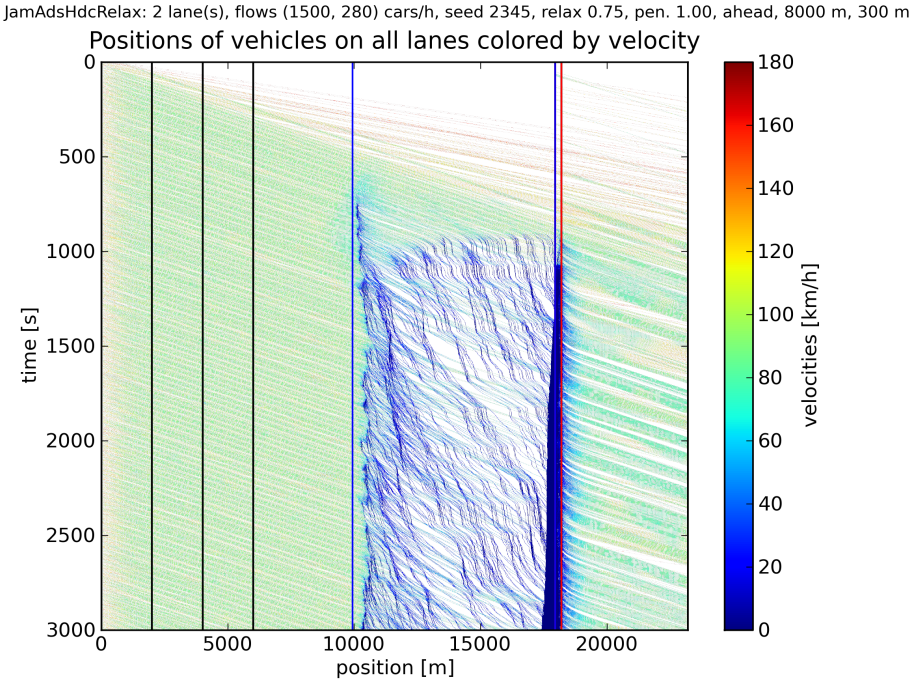
$$r_{\text{spec}} = \begin{cases} 0 & \text{if } t_{\text{avg headway}} \geq t_{\text{thresh}} \\ 1 - \frac{t_{\text{avg headway}}}{t_{\text{thresh}}} & \text{if } t_{\text{avg headway}} < t_{\text{thresh}}. \end{cases} \tag{8.9}$$

Thus, the special relax-factor r_{spec} is larger, the smaller the average time headway $t_{\text{avg headway}}$ is and there is a maximum time headway t_{thresh} above which the strategy is completely turned off.

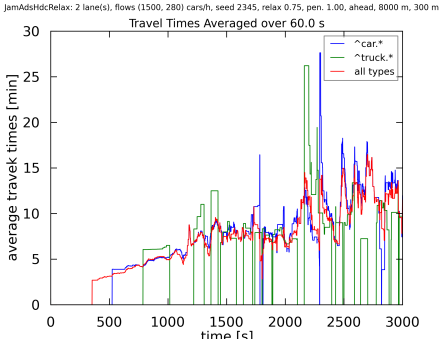
One way to compute the average time headway $t_{\text{avg headway}}$ is to sum up the individual time headways of the equipped vehicles ahead within the HDC (simply denoted as {vehicles}) and divide it by their number

$$t_{\text{avg headway}} = \frac{1}{|\{\text{vehicles}\}|} \sum_{c \in \{\text{vehicles}\}} \frac{g_c}{v_c} \quad , \tag{8.10}$$

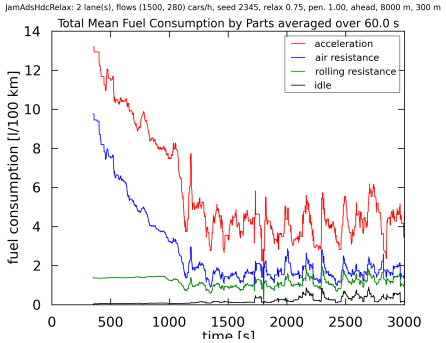
with g_c as current gap and v_c as current velocity of vehicle c. We call this the correct way, because this is the arithmetic average of the time headways of the



(a) Positions plot



(b) Travel times in evaluation section (between the blue vertical lines above)



(c) Mean fuel consumptions in evaluation section (between the blue vertical lines above)

Figure 8.18.: Relax-strategy braking with a general relax-factor $r_{\text{gen}} = 3/4$ and with observed area extended into on-ramp section.

individual vehicles. However, we also tested an average time headway $t_{\text{avg headway}}$ defined as

$$t_{\text{avg headway}} = \frac{\sum_{c \in \{\text{vehicles}\}} g_c}{\sum_{c \in \{\text{vehicles}\}} v_c} \quad , \tag{8.11}$$

which we denote as "broken", because it is in general different from the arithmetic average of individual time headways. Only if all vehicles have the same velocity both average time headways are equal.

Simulations revealed that the correct average time headway of Equation (8.10) is not very effective, but the average time headway according to Equation (8.11) is in general more effective than the relax-strategy depending on braking vehicles. Note, that as already mentioned Figure 8.17 exhibits plots of a simulation run that behaved better than other runs with similar configurations.

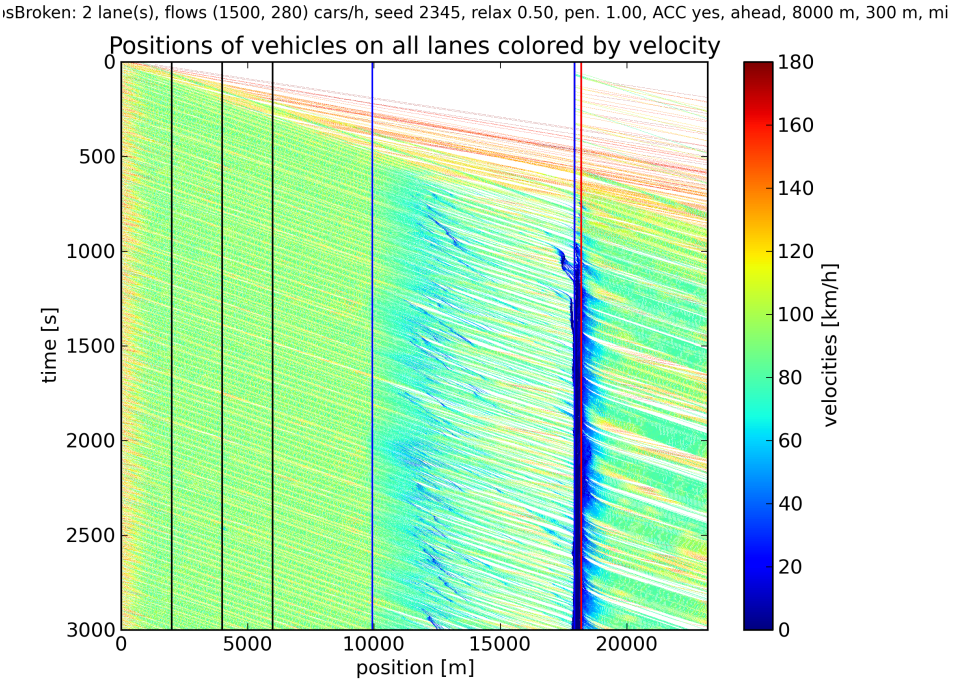
For this time-headway-dependent strategy we need to know the gap to the predecessor of each equipped vehicle even if the predecessor is not equipped. To achieve this we assume that all equipped vehicles are equipped with ACC (Section 4.6) as well.

Simulations also showed that the extension of the observed area into the on-ramp section is a major improvement for this strategy. The extension is done in the same way as for the relax-strategy depending on braking described in the last section.

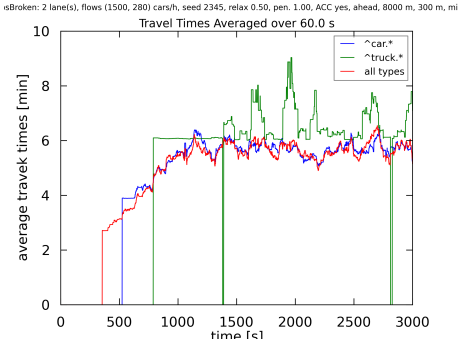
Figure 8.19 contains plots of a simulation run with time-headway relax-strategy including all described improvements. The positions plot in Figure 8.19a shows that traffic jam is strictly located at the on-ramp. Within the remaining HDC-area there are some minor decelerations, but no congestion.

The travel times (Figure 8.19b) are nearly as good as with continuous Jam-ADS, but without the artificial reduced inflow of continuous Jam-ADS. The mean fuel consumption within the evaluation section (Figure 8.19c) is on average better than the fuel consumptions of all other presented simulation runs with reasonable results.

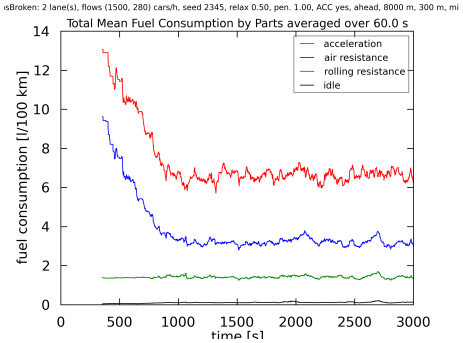
We assume that a strategy depending on time headways is preferable, because it jointly considers gaps and velocities ahead, while all other strategies we presented only depend on the velocities of the vehicles ahead.



(a) Positions plot



(b) Travel times in evaluation section (between the blue vertical lines above)



(c) Mean fuel consumptions in evaluation section (between the blue vertical lines above)

Figure 8.19.: Time-headway relax-strategy with a general relax-factor $r_{\text{gen}} = 3/4$ and with observed area extended into on-ramp section.

9. Analytical Results

This chapter discusses Jam-ADS from a mathematical point of view. First, we look at the state space of the Krauß model and investigate the characteristics of the trajectory of a car in the state space without and with Jam-ADS. Then, we give a lower bound on the optimal value of λ. This is followed by an investigation of the reaction of a chain of vehicles without random deceleration to a single perturbation of the second vehicle. Finally, we linearize the Krauß model, apply tools of control theory of engineering to it and prove that Jam-ADS makes traffic flow stable.

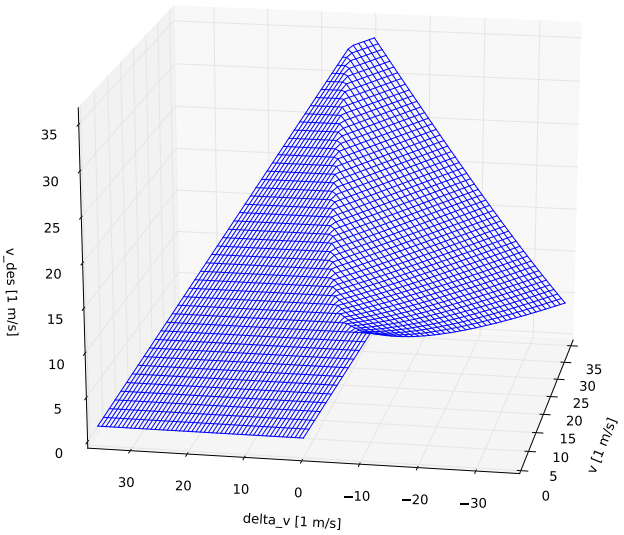
9.1. The state space of the Krauß model

9.1.1. Without Jam-ADS

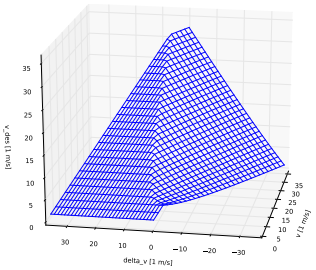
All examples in this section refer to a scenario with a circular road populated with 40 vehicles/km with a length of 5 m each; thus the average gap is $g = 20\,\mathrm{m}$. The time constants are $\tau = \Delta t = 1\,\mathrm{s}$, and the maximum acceleration and deceleration are assumed to be $a = 1.5\,\mathrm{m/s^2}$ and $b = 4.5\,\mathrm{m/s^2}$. From these values and the expression for v_{safe} of the Krauß model (Equation (3.9)) follows an equilibrium velocity without random deceleration of $v_0 = v_{\mathrm{pred}} = g/\tau = 20\,\mathrm{m/s}$.

The diagrams in Figure 9.1 show the desired velocity v_{des} that the Krauß model assigns to a vehicle in the next step, before applying random deceleration with different gaps to the predecessor according to Equation (3.10). The independent values at the bottom are (from front to rear) the velocity of the observed vehicle itself and (from right to left) the velocity difference to the predecessor. The latter is in reverse order to make the interesting parts of the diagram better visible. Projecting the surface down to the independent value plane results in a parallelogram, because the velocity difference is limited such that there are no negative velocities or velocities larger than the total maximum velocity of 36 m/s.

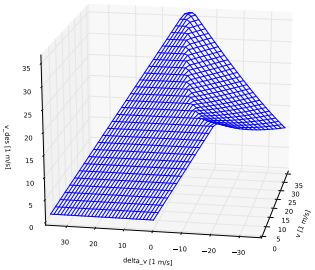
The surface consists of a plane area on the left and a curved area on the right. These correspond to free maximum acceleration and to v_{safe} of the Krauß model, which are two of the three elements in the minimum of v_{des} (Equation (3.10)). If $v_{\max}$ is smaller than the velocity belonging to the top ridge of the surface, there is a plateau on top corresponding to $v_{\max}$, as can be seen slightly in Figure 9.1c. We call these areas the *acceleration regime*, v_{safe} *regime* and v_{max} *regime*.



(a) Medium gap: $g = 20\,\mathrm{m}$



(b) Small gap: $g = 5\,\mathrm{m}$



(c) Large gap: $g = 50\,\mathrm{m}$

Figure 9.1.: Desired velocity v_{des} (before applying random deceleration) for the next step, without Jam-ADS, in dependency of the vehicle's own velocity and the velocity difference to the predecessor for $\Delta t = \tau = 1\,\mathrm{s}$, $a = 1.5\,\mathrm{m/s^2}$, $b = 4.5\,\mathrm{m/s^2}$, and $v_{\mathrm{max}} = 36\,\mathrm{m/s}$. Each diagram shows a different gap to the predecessor.

The acceleration regime's plane hovers exactly $a\Delta t$ above the plane defined by $v(t+1) = v(t)$, which we call the *plane of stationary velocity*.

Increasing the gap to the predecessor (Figure 9.1c) grows the acceleration regime and shrinks the safe-velocity regime, until it no longer imposes a restriction, which corresponds to free traffic. Conversely, decreasing the gap (Figure 9.1b) makes the safe-velocity regime grow at the expense of the acceleration regime.

Without random deceleration a vehicle moves along a trajectory on the surface provided we consider the changes of the surface with changing gaps. With random deceleration the trajectory nodes of the simulation steps are up to $\varepsilon a\Delta t$ (see Equation (3.11)) below the surface. The height of each trajectory node is the v value of the next step.

If we run a simulation starting with equidistant vehicles on a circular road and without random deceleration, the trajectory stays on the vertical plane defined by $\Delta v = 0$. Starting from standing vehicles, the trajectory moves up the acceleration surface in steps of $a\Delta t$, until it reaches the safe-velocity regime. With random deceleration and low density, the trajectory is essentially the same, but fluctuating around the trajectory without random deceleration.

The effect of the random deceleration on the trajectory can be seen in Figure 9.2. It is a projection of the v_{des} surface of Figure 9.1 onto the base plane. The x-axis is also oriented in reverse for easier reference to the 3D surface diagrams. The red circle marks a velocity state before applying random deceleration. The random deceleration independently reduces the velocities of our vehicle and the predecessor up to $\varepsilon a\Delta t$ (Equation (3.11)), moving our vehicle's state to a random point within the parallelogram with uniform probability for each point. As explained earlier in this thesis, we always set $\varepsilon = 1$.

Figure 9.3 and the following Figures 9.4 to 9.8 show the same projection of the v_{des} surface of Figure 9.1 onto the bottom (Δv, v) plane. The gray triangular areas on the top left and bottom right are invalid states, because in these areas the velocity $v+\Delta v$ of the predecessor would either be negative or above $v_{\max} = 36\,\text{m/s}$. The colored lines extend into these areas only for technical reasons.

The cyan-colored curved line denotes the border between the acceleration regime below and the v_{safe} regime above. The magenta-colored line closely above the cyan-colored line indicates the border between states in which our vehicle can accelerate (below) and states in which it must decelerate (above). States on the line must keep their velocity. The magenta line is the projection of the intersection of the plane of stationary velocity with the v_{des} surface.

Above the magenta-colored line is the region in which v_{safe} is smaller than the current velocity v. The strip between the cyan and the magenta lines are states with $v < v_{\text{safe}} < v + a\Delta t$, hence the vehicle can accelerate in these states, but with less than maximum acceleration a. Both lines are parabolas with a non-vertical axis of symmetry.

The dashed blue horizontal line denotes the assumed average velocity v_{avg}. Fi-

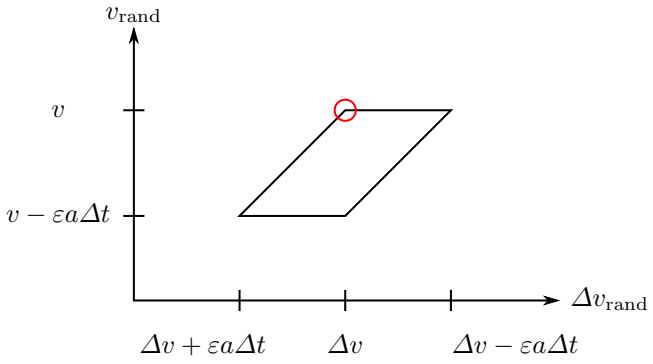


Figure 9.2.: The set of possible values after random deceleration. Suppose that before applying random deceleration, a vehicle has velocity v and velocity difference Δv to its predecessor as marked by the red circle. Then random deceleration moves it to a random point within the parallelogram.

nally, the little parallelogram is the parallelogram explained in Figure 9.2. In all diagrams of this type its top left corner—the state before random deceleration—corresponds to the state after applying the Krauß model to the initial state ($\Delta v = 0$ and $v = v_{\text{avg}}$; the state at the center of the dashed line) but before random deceleration. As explained, the points within the parallelogram are the possible new states after random deceleration.

In Figure 9.3 v_{avg} is set to this equilibrium velocity v_0. Therefore the magenta line, on which vehicles keep their velocity, intersects the dashed v_{avg} line at $\Delta v = 0$. This is also the upper left corner of the parallelogram, because there is no change of the state before random deceleration. However, the parallelogram sits completely below the v_{avg} line, so with random deceleration all vehicles become slower, which decreases v_{avg}. In addition, points within the parallelogram above the magenta line have to decelerate in the next simulation step, because $v_{\text{safe}} < v$. Thus, this state is not stable with random deceleration.

One might guess from this state, that all vehicles decelerate until v_{avg} divides the parallelogram into equal halves, as shown in Figure 9.4, because starting from that state after the next simulation step and including random deceleration, we end above or below v_{avg} with equal probability. However, states around the upper right corner of the parallelogram above the magenta line have a v_{safe} below the current velocity, so they are forced to decelerate further during the next simulation step.

Local changes in the parameters and looking at the parallelograms of further

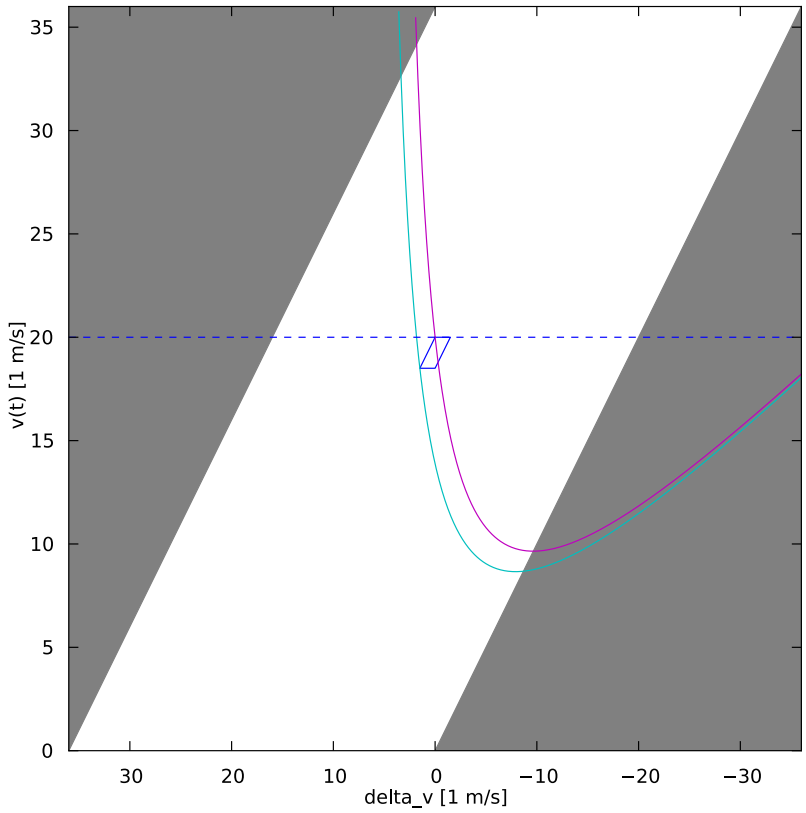


Figure 9.3.: The space of vehicle states for a gap $g = 20\,\mathrm{m}$. This gap corresponds to the average gap at a density of 40 vehicles/km with a vehicle length of 5 m. The average velocity (blue dashed line) is set to $v_{\mathrm{avg}} = 20\,\mathrm{m/s}$ corresponding to the equilibrium velocity without random deceleration.

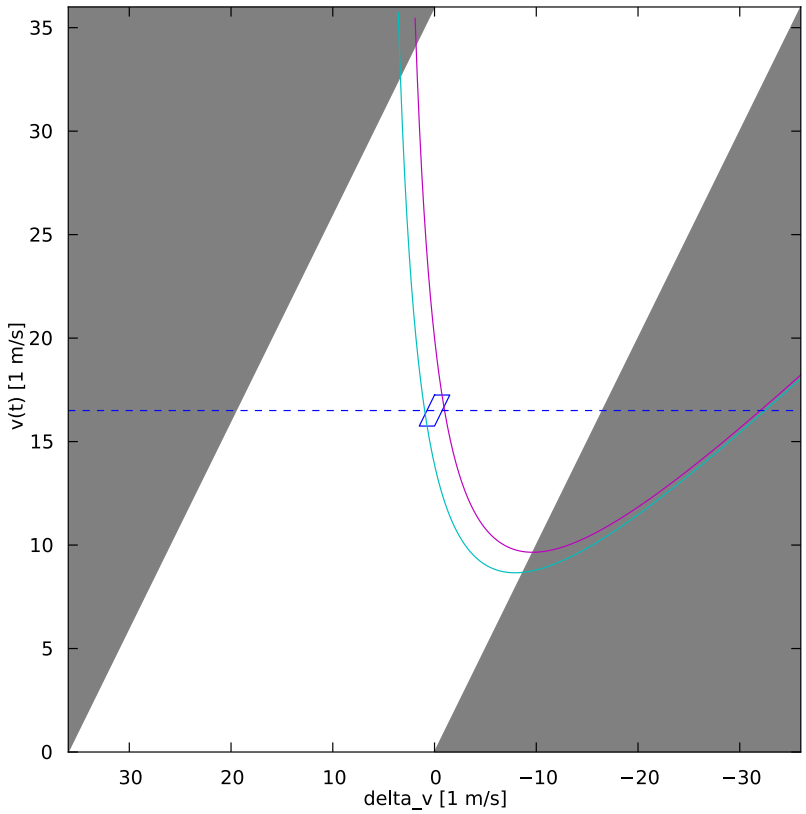


Figure 9.4.: Figure 9.3 with the average velocity (blue dashed line) set to $v_{\mathrm{avg}} = 16.5\,\mathrm{m/s}$ such that the parallelogram becomes cut in equal halves, approximately.

steps revealed that if random deceleration falls into the region around the upper right corner of the parallelogram, the deceleration enforced by v_{safe} is likely to be more than the maximum acceleration a. This particularly makes the gap to the following vehicle much smaller, such that the follower has to decelerate even more. This continues until the lower part of the magenta line is reached. Note that the decreasing gap makes the cyan and magenta lines fall as well.

As presented in Chapter 7 (see also Figure 6.2 for an example), a circular road with high vehicle density displays massive traffic jams with free traffic sections between them. The corresponding parallelogram of a simulated vehicle in the traffic jam can be seen in Figure 9.5, and a simulated vehicle in free traffic in Figure 9.6. So the state of some vehicles decreases along the magenta line down to the apex (with the line itself moving down because of the decreased gap). This empties more space to the remaining vehicles to allow them to reach the maximum velocity.

9.1.2. With Jam-ADS

With Jam-ADS the part of the desired velocity above v_{avg} is reduced by the factor λ. Vehicles below ($v_{\text{des}} < v_{\text{avg}}$) are not affected by Jam-ADS, because for these we have $v_{\text{Jam-ADS}} > v_{\text{des}}$ (Equation (5.1)), thus the minimum of Equation (5.2) makes $v_{\text{rec}} = v_{\text{des}}$. Figure 9.7 shows the resulting v_{rec} of the next simulation step, depending on the (Δv, v) state.

Figure 9.8 shows the effect of Jam-ADS on the projection of the v_{rec} surface (Figure 9.7) to the base plane of (Δv, v) states. The projection of the border between the regimes (cyan-colored line) is not affected, because the compression is purely vertical. However, the magenta line separating acceleration and deceleration changes. To understand this, we imagine the plane of stationary velocity in Figure 9.7 that is parallel to the lower left planar area, but a little below ($a\Delta t = 1.5\,\text{m/s}$ to be exact), such that it intersects the Δv axis at the bottom front. Its intersection with the surface is the magenta line of Figure 9.8.

The plane of stationary velocity intersects the acceleration regime above the crease on the v_{avg} level, which is shown as a dash-dotted blue line in Figure 9.8.[1] Hence, the magenta line equals the blue dash-dotted line within the acceleration regime. The intersection of the plane of stationary velocity with the upper part of the acceleration regime above v_{avg} defines a new dynamic maximum velocity, depending on λ and v_{avg}. This maximum Jam-ADS velocity is defined by

$$v_{\text{max, Jam-ADS}} = \lambda(v_{\text{max, Jam-ADS}} + a) + (1 - \lambda)v_{\text{avg}} \quad . \tag{9.1}$$

[1] This makes the term "acceleration regime" inappropriate above the intersection, but we stick to it for easier reference.

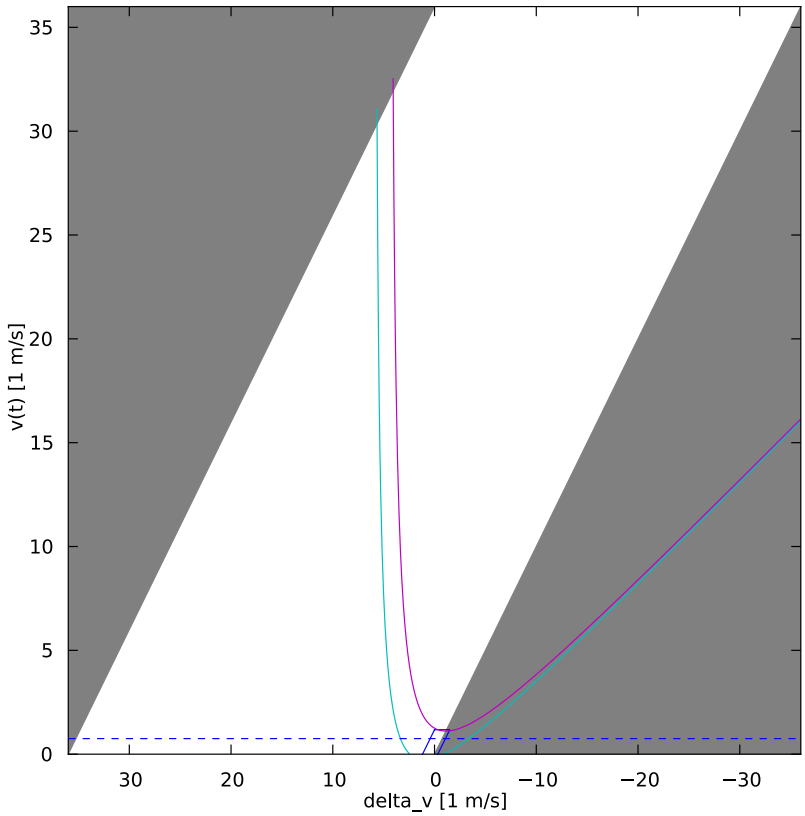


Figure 9.5.: Parallelogram of a vehicle inside of a traffic jam.

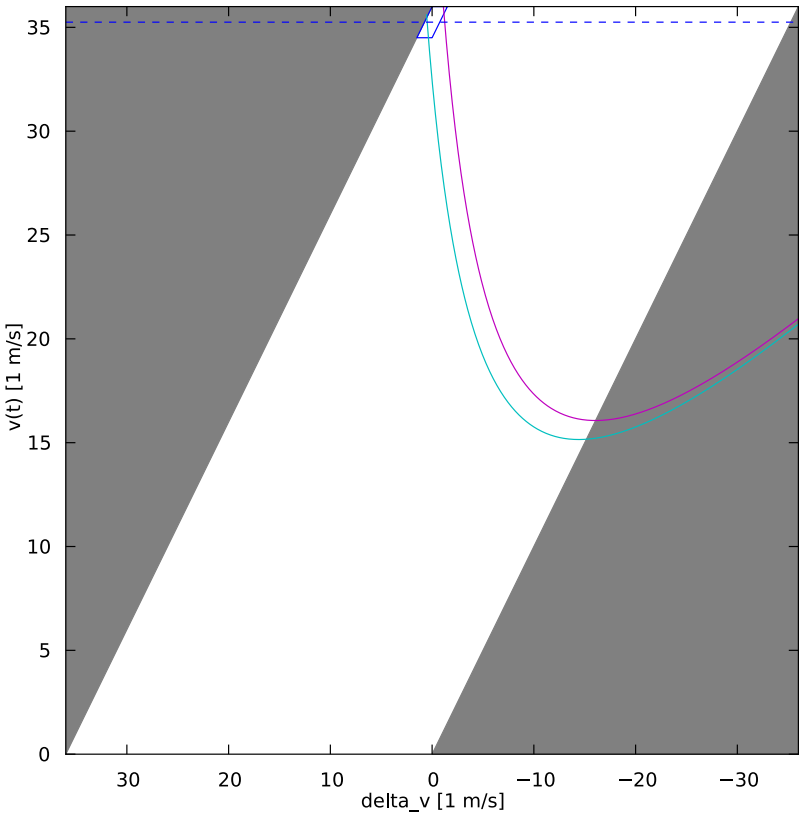
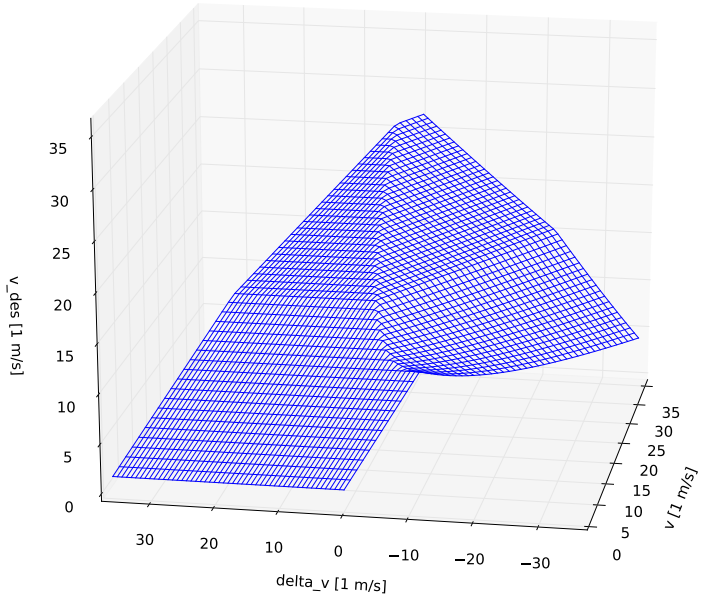
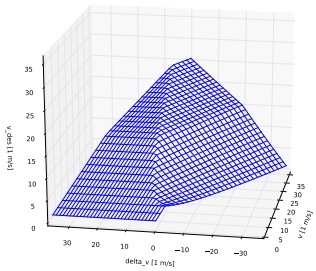


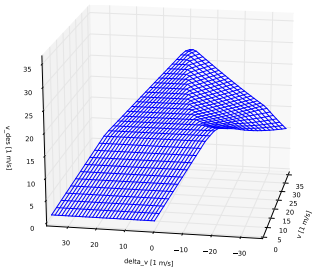
Figure 9.6.: Parallelogram of a vehicle in free traffic.



(a) Medium gap: $g = 20\,\mathrm{m}$



(b) Small gap: $g = 5\,\mathrm{m}$



(c) Large gap: $g = 50\,\mathrm{m}$

Figure 9.7.: Recommended velocity by Jam-ADS, v_{rec}, (before applying random deceleration) in dependency of our vehicle's own velocity and the velocity difference to the predecessor. All configuration values are the same as in Figure 9.1. Again, each diagram displays a different gap to the predecessor.

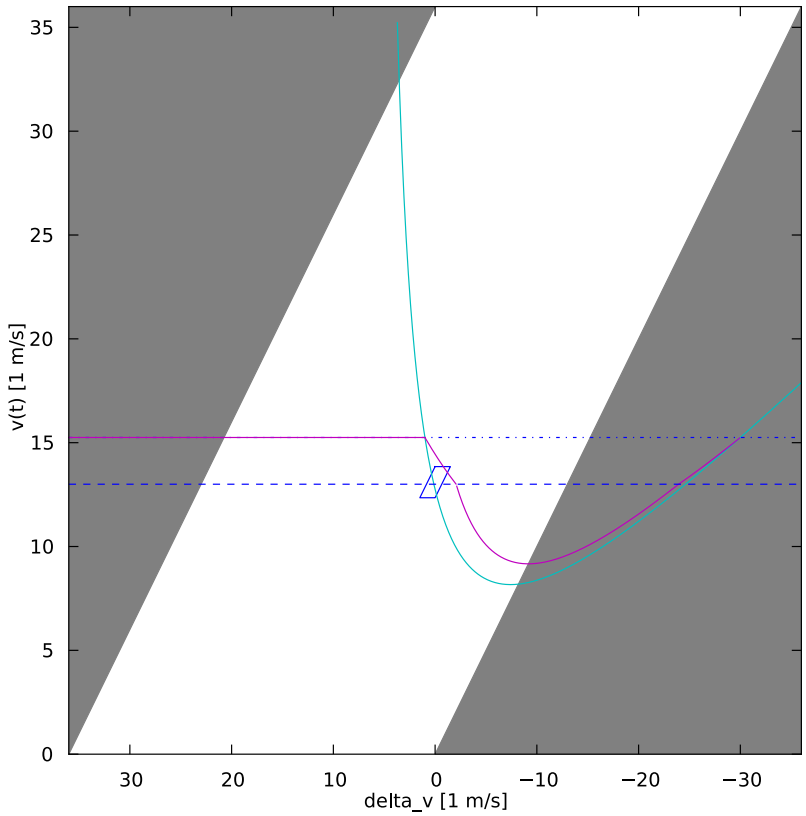


Figure 9.8.: Parallelogram with Jam-ADS.

Solving it for $v_{\text{max, Jam-ADS}}$ results in

$$v_{\text{max, Jam-ADS}} = v_{\text{avg}} + \frac{\lambda}{1-\lambda} a \quad . \tag{9.2}$$

Simulations revealed that the Jam-ADS equilibrium is close to the vertex at which the four parts of the surface intersect, but not exactly on it. This allows us to approximate the Jam-ADS equilibrium velocity v_0, because Δv must be zero at the equilibrium and the equilibrium gap g_0 must be close to the average gap, depending on the known density ρ and vehicle length ℓ. The vertex is at the intersection of the acceleration and the v_{safe} regimes for the given gap g_0:

$$v_0 + a = v_{\text{safe}}(v = v_0, \Delta v = 0, g = g_0 = 1/\rho - \ell) \quad . \tag{9.3}$$

Solving it for v_0 yields

$$v_0 = \frac{g_0 - a\Delta t\tau}{\frac{a\Delta t}{b} + \tau} \quad . \tag{9.4}$$

In our standard configuration ($g_0 = 20\,\text{m}$, $a = 1.5\,\text{m/s}^2$, $\tau = \Delta t = 1\,\text{s}$, $b = 4.5\,\text{m/s}^2$), this results in $v_0 = 13.875\,\text{m/s}$. Simulations show $v_0 \approx 13.0\,\text{m/s}$.

A reason for this deviation is that random deceleration causes a non-singular distribution of the gap. Figure 9.9 shows the gap distribution of a simulation with the standard configuration. The distribution is not symmetric, because gaps can grow much more over the average gap than they can drop below it. Therefore, if a vehicle happens to have a large gap, there must be several vehicles with a slightly smaller-than-average gap to compensate for this. Another reason for the asymmetry is that it takes longer for a large gap to drop back to the average than it takes to grow a small gap back to average, because a small gap enforces deceleration for the sake of safety. This quickly regrows the gap, while shrinking a large gap requires acceleration, which is limited by maximum acceleration a.

If we insert the most probable gap $g_0 = 18.7\,\text{m}$ from Figure 9.9 into Equation (9.4), we get $v_0 = 12.9\,\text{m/s}$, which is closer to the measured value $v_0 \approx 13.0\,\text{m/s}$.

Although the gaps depend on accumulation of velocity differences to the predecessor, the asymmetry of the gap distribution cannot entirely be derived from that. Figure 9.10 shows the probability distribution function of the velocity difference, $p_1(\Delta v)$, after one simulation step, provided it was zero before. It has a variance σ^2 of

$$\begin{aligned} \sigma^2 &= \int_{-\infty}^{\infty} \Delta v^2 p_1(\Delta v) d\Delta v \\ &= \frac{1}{6}(a\Delta t)^2 \quad . \end{aligned} \tag{9.5}$$

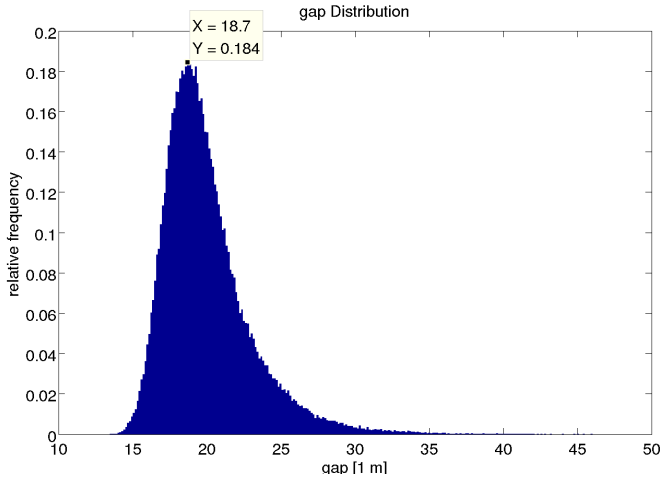


Figure 9.9.: Gap distribution of a simulation with Jam-ADS.

According to the central limit theorem, this converges to a normal distribution after infinitely many steps, denoted as $p_\infty(\Delta v)$:

$$\begin{aligned} p_\infty(\Delta v) &= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\Delta v}{\sigma}\right)^2} \\ &= \sqrt{\frac{3}{\pi}} \frac{1}{a\Delta t} e^{-3\left(\frac{\Delta v}{a\Delta t}\right)^2} \quad . \end{aligned} \tag{9.6}$$

Figure 9.11 shows the Δv distribution produced by simulation (blue bars) and this p_∞ (black curve). They are similar but clearly different. Prolonging the simulation does not produce a significant change, hence we do not expect the measured distribution to converge to p_∞. However, we can see that the measured velocity difference distribution is symmetric, opposed to the gap distribution in Figure 9.9, above. The difference between the measured and the theoretically derived distribution indicates that the variance of the velocity difference of successive vehicles is not only determined by the random deceleration. Further research should clarify this and derive the gap distribution.

9.2. Lower bound of λ

Because of random deceleration, a value of λ in Jam-ADS that is too small causes all vehicles to stop or come to a crawl. To explain this, we first examine the case $\lambda = 0$, i.e., we recommend the average velocity v_{avg} (Equation (5.1)) of the vehicles

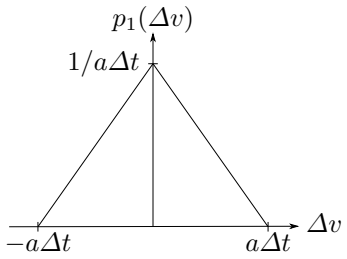


Figure 9.10.: Probability distribution of velocity difference because of random deceleration after one simulation step.

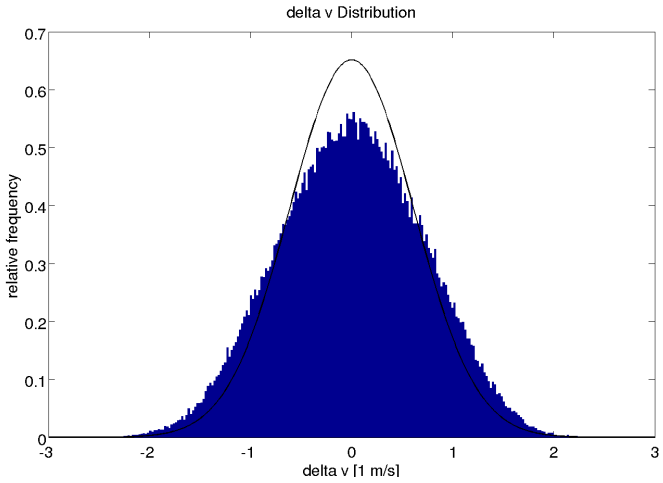


Figure 9.11.: Δv distribution with Jam-ADS (blue bars) and theoretical limit p_∞ (black curve).

ahead without considering the driver's own desired velocity v_{des} (Equation (3.10)). In this case every slight decrease of a vehicle's velocity due to random deceleration also decreases v_{avg} for the vehicles upstream, so they have to decelerate as well. A possible reacceleration could not take place, because acceleration only affects the desired velocity v_{des}, which we do not consider. Thus, with $\lambda = 0$ random deceleration makes all vehicles stop after a finite time.

Next, we consider an arbitrary $\lambda > 0$. In addition, we assume $\lambda < 1$, because $\lambda = 1$ turns off Jam-ADS. We suppose the system to be initially in an equilibrium state with all vehicles having an equilibrium velocity of v_0 due to v_{safe}. As explained in Section 3.2.5, the expectation value of the random deceleration is $\varepsilon a \Delta t/2$, which together with our setting of $\varepsilon = 1$ results in an expectation value of $a\Delta t/2$. Additionally, we do not consider a further deceleration due to v_{safe}. Thus, the expectation value of the average velocity at time step k under Jam-ADS, $v_{\text{avg}}^{(k)}$, follows the recursion

$$v_{\text{avg}}^{(k+1)} = \lambda \min(v_{\text{avg}}^{(k)} + a, v_0) + (1-\lambda) v_{\text{avg}}^{(k)} - \frac{1}{2}a \quad , \tag{9.7}$$

because the minimum is the v_{des} of the Krauß model to which we apply Jam-ADS and the mean random deceleration.

We start the recursion with the equilibrium velocity, $v_{\text{avg}}^{(0)} = v_0$, hence during the initial time steps $v_{\text{avg}}^{(k)}$ is close enough to v_0 such that $\min(v_{\text{avg}}^{(k)} + a, v_0) = v_0$. As long as this is the case, the recursion resolves to

$$v_{\text{avg}}^{(k+1)} = v_0 - \frac{1}{2}a \sum_{i=0}^{k} (1-\lambda)^i \quad . \tag{9.8}$$

With our assumption $0 < \lambda < 1$ the power series converges to

$$\sum_{i=0}^{k} (1-\lambda)^i \xrightarrow{k\to\infty} \frac{1}{\lambda} \quad . \tag{9.9}$$

Thus for $\lambda \geq 1/2$ Equation (9.8) holds for all time steps k, because the power series monotonically increases.

For $\lambda < 1/2$ there exists a minimal k_0 with $v^{(k_0)} < v_0 - a$. Thus

$$\forall_{k \geq k_0} : \min(v_{\text{avg}}^{(k)} + a, v_0) = v_{\text{avg}}^{(k)} + a \tag{9.10}$$

Hence, in this case for $k \geq k_0$ the recursion resolves to

$$v_{\text{avg}}^{(k)} = v_{\text{avg}}^{(k_0)} + (k - k_0)(\lambda - \frac{1}{2})a \quad , \tag{9.11}$$

which diverges to

$$v_{\text{avg}}^{(k)} \xrightarrow{k\to\infty} -\infty \quad . \tag{9.12}$$

The Krauß model does not allow negative velocities, so with $\lambda < 1/2$ the velocity drops until all vehicles stop.

We confirmed these results by simulations. In case of $\lambda = 1/2$, the system is in an unstable equilibrium. We have seen that λ must fulfill $\lambda \geq 1/2$ to make Jam-ADS stable. In the border case $\lambda = 1/2$, velocity does not necessarily drop to zero, but can also not recover back to equilibrium. Thus, we usually set $\lambda = 0.6$ if we need a concrete value for λ, which is small enough to make Jam-ADS effective but not too close to the border $\lambda = 1/2$.

9.3. Single perturbation

9.3.1. Modeling approach

Another approach to mathematically analyze the effect of Jam-ADS is to study the reaction of a chain of vehicles to a single perturbation without any randomness. To model this, we assume that initially all vehicles have the same gap g_0 to their respective predecessor and all move with the same equilibrium velocity v_0. In addition, we chose v_0 such that $v_0 \ll v_{\max}$, because we are interested in dense traffic, that means all vehicles are definitely slower than their maximum velocity. Thus, the common equilibrium velocity is given by v_{safe}, because in equilibrium and without randomness there is no acceleration.

Setting v_{safe} and v_{pred} to v_0 in the formula for v_{safe} (Equation (3.9)) leads to the equilibrium condition

$$g_0 = v_0 \, \tau \quad . \tag{9.13}$$

Thus, the equilibrium gap and velocity have a fixed relation.

To see a relaxation back to the given equilibrium, we perturb the second vehicle in the chain, while the first vehicle holds the equilibrium velocity forever. The initial velocity of the second vehicle is set to $v_0 - p$ and the initial gap to $g_0 - p\tau$. These values fulfill the equilibrium relation Equation (9.13) as well.

Table 9.1 gives an overview over the parameter values chosen for every plot in this section. The table does not contain the vehicle length, because we only look at the gaps between the vehicles without considering their absolute positions, thus the vehicle lengths are irrelevant. The time development of velocity and gap to the predecessor of the perturbed second vehicle can be seen in Figure 9.12.

Table 9.1.: Configuration of the following plots of time development of a single perturbation. The values correspond to the values in Table 3.1, which are known to reproduce realistic traffic jams.

Value	Symbol	Parameter
Time step length	Δt	1 s
Reaction time	τ	1 s
Maximum velocity	$v_{\max}$	36 m/s
Maximum deceleration	b	4.5 m/s^2
Maximum acceleration	a	1.5 m/s^2
Equilibrium velocity	v_0	10 m/s
Equilibrium gap	g_0	10 m
Initial perturbation	p	1.5 m/s
Jam-ADS λ	λ	0.6
Jam-ADS number of averaged vehicles		5

We can see that both velocity and gap overshoot during relaxation to the equilibrium values. In fact, there is a second small oscillation below the equilibrium line around 10 s, which is too small to be visible. If we decrease the equilibrium values of the velocity v_0 and gap g_0, a damped oscillation becomes visible. We prove this analytically, below.

9.3.2. Observations without Jam-ADS

Figures 9.13 and 9.14 show the effects of the perturbation further down the chain of vehicles. Note that these plots cover the whole velocity space from zero to $v_{\max}$, as well as the respective gap space, opposed to the small limits of the plot of the second vehicle in Figure 9.12. We see, that the small perturbation of the second vehicle causes large oscillations further down the vehicle chain. Until the limits of zero and $v_{\max}$ are hit, the oscillations are similar to a wave packet (Figure 9.13), although they are not completely symmetric. When the limits of zero and $v_{\max}$ are encountered, the oscillations degenerate to less regular shapes (Figure 9.14). Note that the gap has no maximum limit, as opposed to the velocity, which does have a maximum limit.

In all cases, the oscillations stop after a finite time and the vehicle returns to the equilibrium velocity and gap.

The time until the perturbation reaches the n^{th} vehicle is always at least n seconds. This is a consequence of the traffic-model step length $\Delta t = 1\,\text{s}$, because it takes at least one time step until a vehicle reacts to its predecessor.

The Krauß model is designed such that a vehicle tries to achieve a desired gap to its predecessor asymptotically, except the maximum velocity prevents that. As

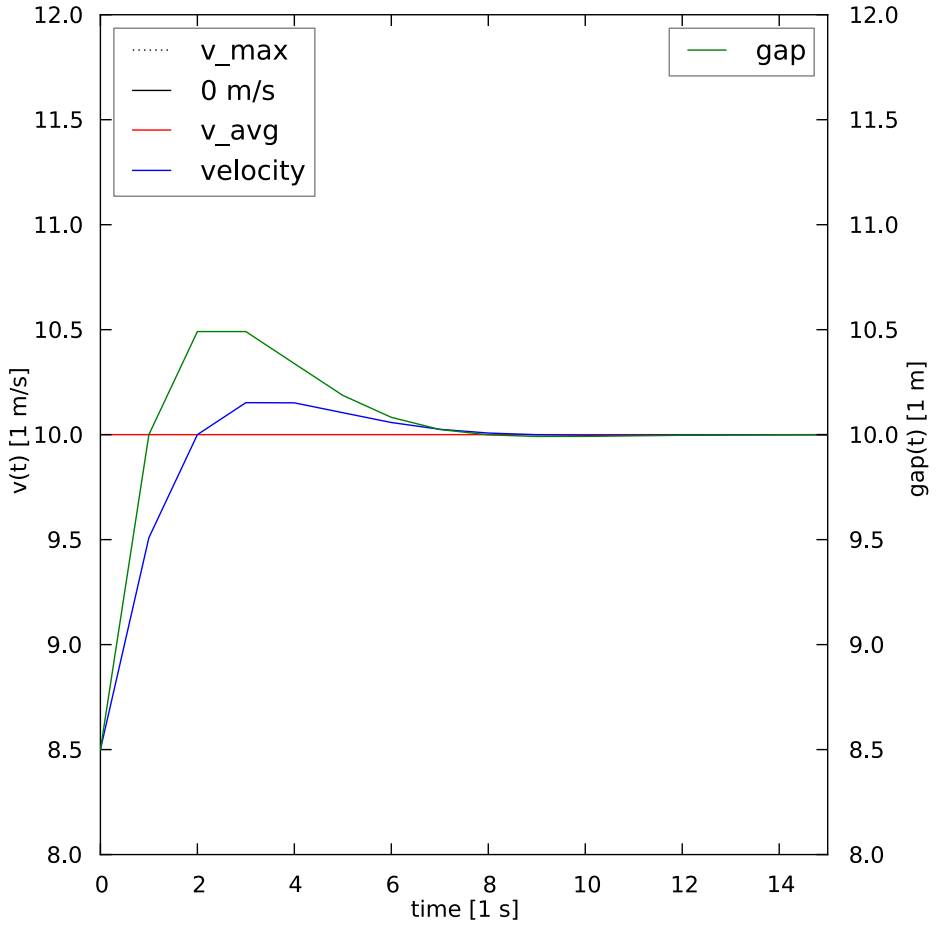


Figure 9.12.: Time development of velocity (blue) and gap to the predecessor (green) of the second vehicle, which is perturbed in the initial time step. Here, the red line of average velocity is fixed to the equilibrium velocity v_0, because without Jam-ADS we do not average over vehicles ahead. The dotted and solid black lines of $v_{\max}$ and zero, noted in the legend, are outside of the plot limits.

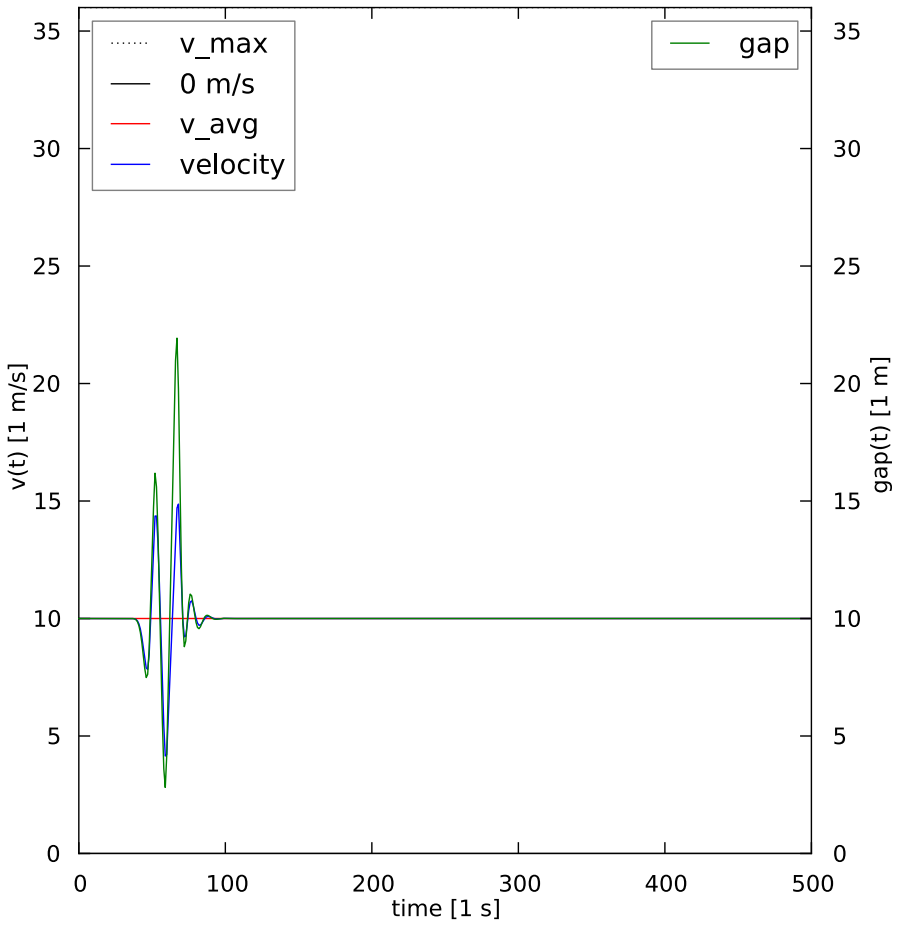


Figure 9.13.: Velocity (blue) and gap to the predecessor (green) of the 35^{th} vehicle in the chain. The dotted and solid black lines of $v_{\max}$ and zero, mentioned in the legend, are hidden by the top and bottom edges of the frame. As opposed to the last plot (Figure 9.12) this and the following plots cover the whole range of allowed velocities and belonging gaps.

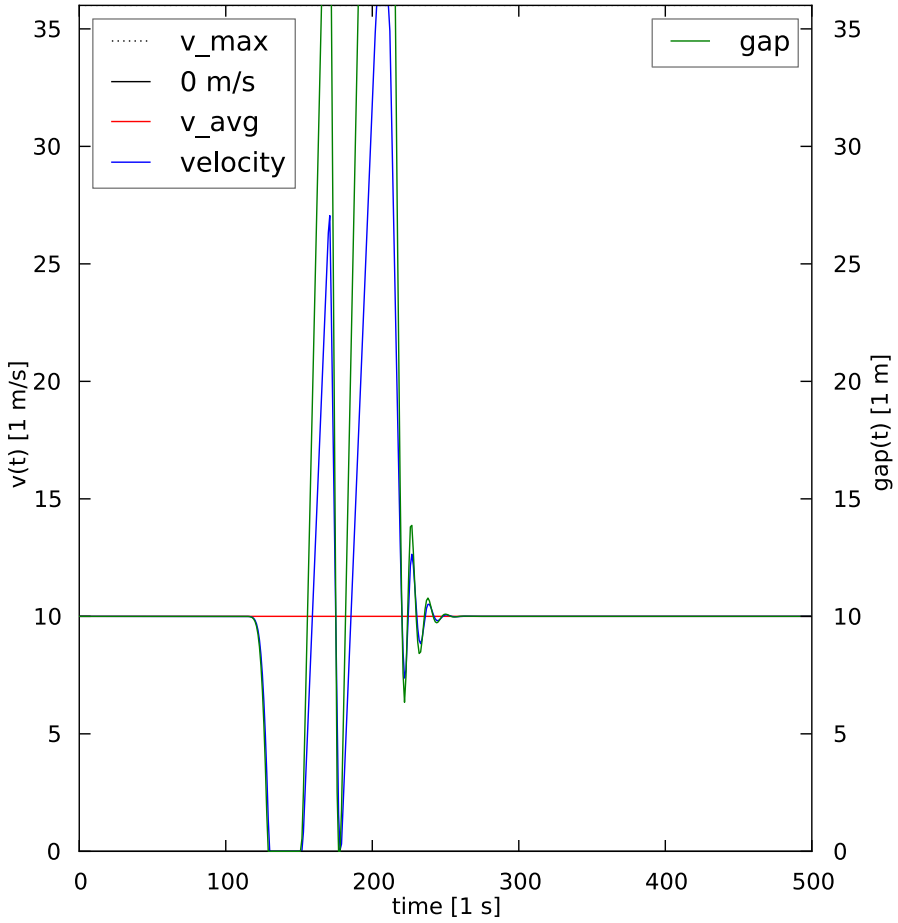


Figure 9.14.: Velocity (blue) and gap (green) of the 100^{th} vehicle in the chain. Both are limited at the bottom edge, but only the blue velocity line is limited at the top edge.

can be seen by Equation (3.9), the desired gap is $v_{\text{pred}}\tau$ and the relaxation time is the denominator $(v+v_{\text{pred}})/2b+\tau$. Relaxation to a desired gap implies adaptation of a vehicle's velocity to the predecessor's velocity.

Therefore, the second vehicle (Figure 9.12) returns to the equilibrium velocity that the first vehicle always keeps. However, the adaptation is such that the correction overshoots before it approaches the first vehicle's velocity and the equilibrium gap. This also applies to the following vehicles, such that with each vehicle along the chain the oscillations grow. In addition, each vehicle returns at some finite time to the equilibrium velocity and gap, but it takes longer for each vehicle further away from the initially perturbed second vehicle. Of course, in reality we have repeating perturbations, so if traffic is dense enough, we never see a complete return to the equilibrium state.

Note that the oscillations of velocity and gap of a vehicle are always synchronized, although the velocity has a maximum while the gap could grow infinitely. This is a consequence of the connected update rules of velocity and gap of the Krauß model.

Another reason for the degeneration of the oscillations when they grow is the limited maximum acceleration. At some point the rising edges of the velocity oscillations reach the limit of maximum acceleration a. This can be seen at the 100^{th} vehicle in Figure 9.14 by its perfectly linear rising edges. This also applies to the predecessor, so for most time of the rising edge both vehicles accelerate with acceleration a while keeping a fixed velocity difference, which makes the gap grow linearly as well.

9.3.3. Further experiments

Perturbation of several leading vehicles

Instead of perturbing only the second vehicle we tried to extend the initial perturbation to the first k vehicles except the leading vehicle, as before.

The effect was an earlier start of the perturbation of each vehicle and a changing shape of the oscillations. The up to k seconds earlier start corresponds to the number of perturbed vehicles, because the last perturbed vehicle was now k vehicles closer to the observed vehicle than in the previous examples with only one perturbed vehicle.

Raising the number of initially perturbed vehicles has no influence on the time when the oscillations become extinct.

Comparison with other car-following models

We described in Section 6.2.1 that SUMO applies a different v_{safe} (Equation (6.1)) than the original Krauß model. Applying that to our single perturbation scenario makes no considerable difference.

Additionally, we tried the Gipps model (Section 3.2.4). In our standard configuration the Gipps model transmits the initial perturbation along the chain of vehicles and it degrades during that. With a higher vehicle density, that is a smaller equilibrium gap g_0 and velocity v_0, the Gipps model also produces heavy oscillations, which also end after finite time for each vehicle. Along the chain of vehicles the amplitude and duration of the oscillations grow, as they do with the Krauß model.

Replace time by vehicle number in plots

So far we presented the time development of the velocity and the gap of one vehicle in the chain. Additionally, we looked at velocities and gaps of all vehicles of the chain at a certain time.

The results were similar oscillations except that they seem to be mirrored on the x-axis. This is, because when a vehicle further down the chain starts to oscillate, another vehicle closer to the beginning of the chain finishes its oscillations. So the chain of vehicles at a certain point in time exhibits the oscillations in reverse.

We can conclude that the temporal and the spatial development of a perturbation is qualitatively the same. This result supports the idea to store the congestion data in geographically fixed HDCs (Section 5.1), because the oscillations seem to take place at a fixed location.

9.3.4. Observations with Jam-ADS

We now turn on Jam-ADS with average-strategy. As opposed to our simulation experiments, we do not have absolute positions now. Hence, v_{avg} in Equation (5.1) is taken over a fixed number of vehicles ahead instead of a certain distance ahead.

Figures 9.15 to 9.17 show the effects on velocity and gap development of the second (Figure 9.15), 35^{th} (Figure 9.16), and 100^{th} (Figure 9.17) vehicle. In every plot the average was taken over the next five vehicles ahead and the convex combination coefficient was set to $\lambda = 0.6$. The red line shows this average over time. Note, that in case of the second vehicle in Figure 9.15 the average could only be taken over the single first vehicle which keeps the constant velocity v_0, thus the red line is straight.

Figure 9.15 shows that the change for the second vehicle was minimal compared to the behavior without Jam-ADS (Figure 9.12). For the first two seconds there was no change of the velocity and for the first three seconds no change of the gap. Then, the relaxation back to the velocity v_0 and gap g_0 of the constantly driving predecessor takes longer.

We show below that Jam-ADS indeed prolongs the relaxation time. Jam-ADS has no influence on the first time steps, because of the minimum in Equation (5.2), which removes the influence of Jam-ADS if the recommended velocity would be

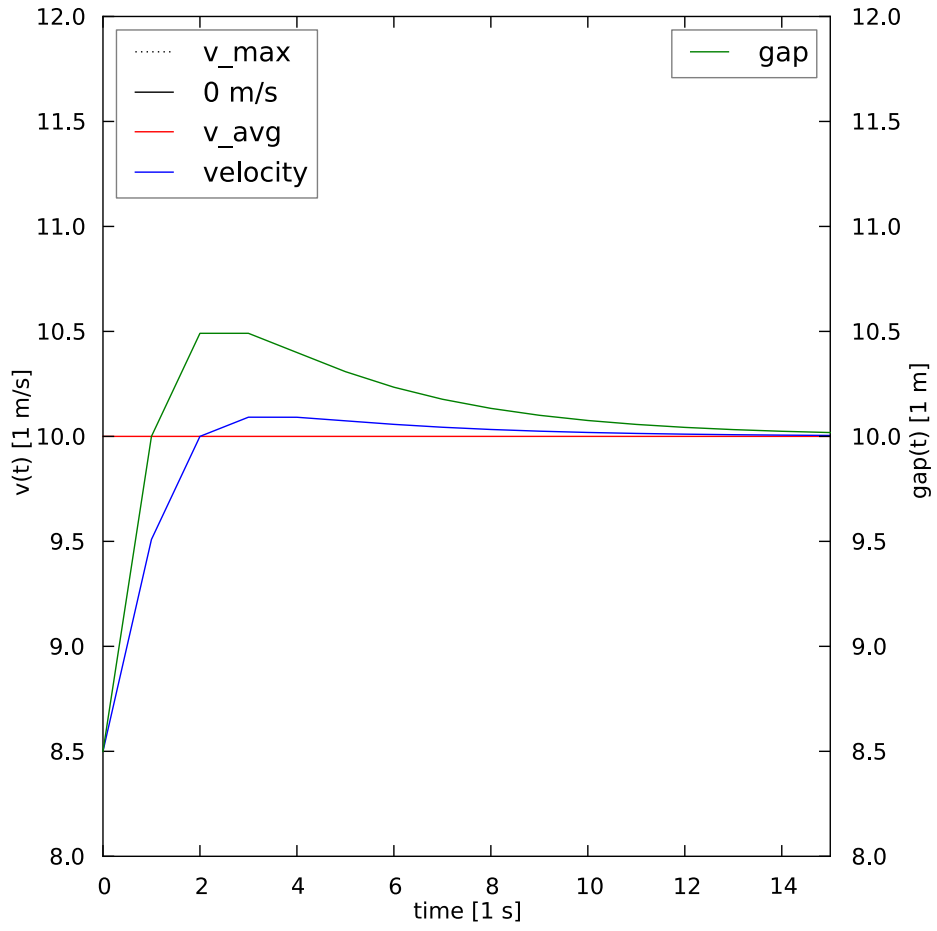


Figure 9.15.: Second (initially perturbed) vehicle with Jam-ADS.

larger than the desired velocity. This is the case if the blue velocity line is below the red line of v_{avg}.

Figures 9.16 and 9.17 show the influence of Jam-ADS on the 35^{th} and 100^{th} vehicle, respectively. The oscillations are nearly completely wiped out except a small perturbation, smaller than the initial perturbation, traveling along the chain of vehicles.

9.4. Linearized Krauß model

9.4.1. Linearization of the Krauß model

It is difficult and could even be impossible to deduce analytical results from the Krauß model because of its non-linear nature. To get some analytical results nevertheless, we linearize the Krauß model.

As in the last section, we assume large enough $v_{\max}$ such that we do not have to consider it as limit. Additionally, we assume that the vehicle density is high enough, such that we are always in the v_{safe} regime, because we are interested in dense traffic.

Hence, we assume that v_{safe} always equals the desired velocity of the next time step. v_{safe} depends on the current velocity v, the predecessor's velocity v_{pred} and the current gap g as variables and on the reaction time τ and maximum deceleration b as constants (Equation (3.9)). Expressing v_{safe} as function of these variables leads to:

$$F_{\text{safe}}(v, v_{\text{pred}}, g) = v_{\text{pred}} + \frac{g - v_{\text{pred}}\tau}{\frac{1}{2b}(v + v_{\text{pred}}) + \tau} \tag{9.14}$$

A Taylor approximation to the first order in v, v_{pred}, and g around $v = v_{\text{pred}} = v_0$ and $g = g_0 = v_0\tau$ produces:

$$\begin{aligned} F_{\text{safe}}(v, v_{\text{pred}}, g) = {} & F_{\text{safe}}(v_0, v_0, g_0) \\ & + \frac{\partial}{\partial v} F_{\text{safe}}(v_0, v_0, g_0)(v - v_0) \\ & + \frac{\partial}{\partial v_{\text{pred}}} F_{\text{safe}}(v_0, v_0, g_0)(v_{\text{pred}} - v_0) \\ & + \frac{\partial}{\partial g} F_{\text{safe}}(v_0, v_0, g_0)(g - g_0) + \dots \\ = {} & v_{\text{pred}} + c(g - v_{\text{pred}}\tau) + \dots \end{aligned} \tag{9.15}$$

with the abbreviation

$$c := \frac{1}{\frac{v_0}{b} + \tau} \quad . \tag{9.16}$$

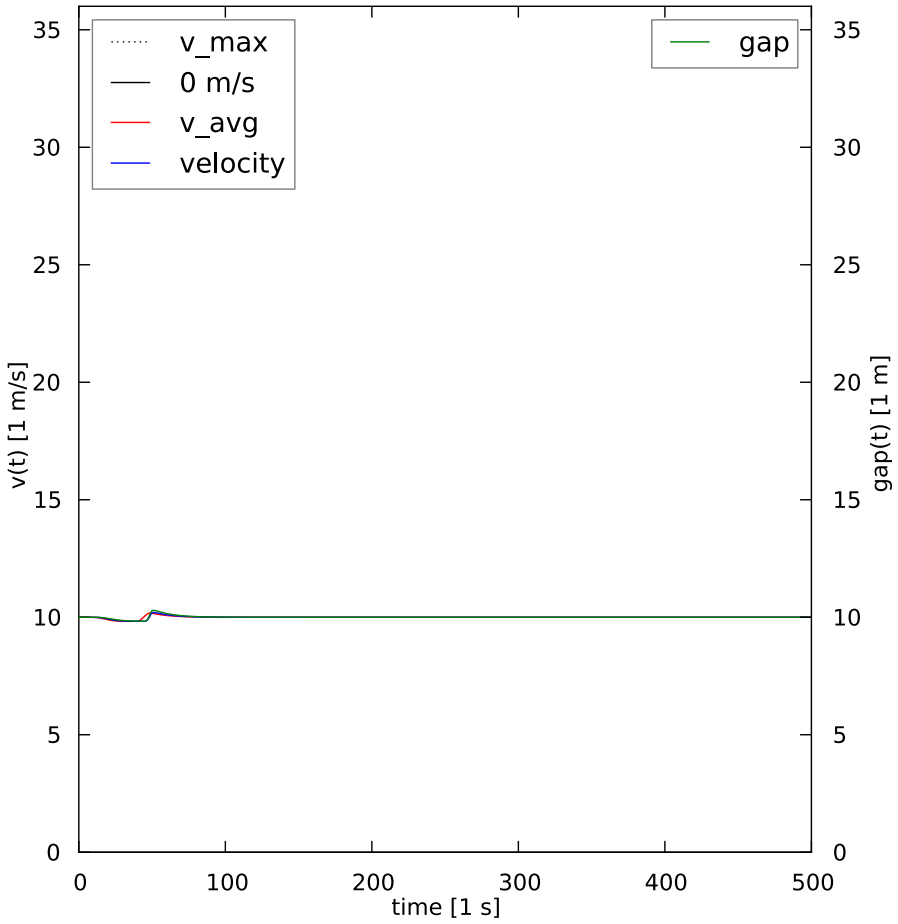


Figure 9.16.: 35^{th} vehicle with Jam-ADS.

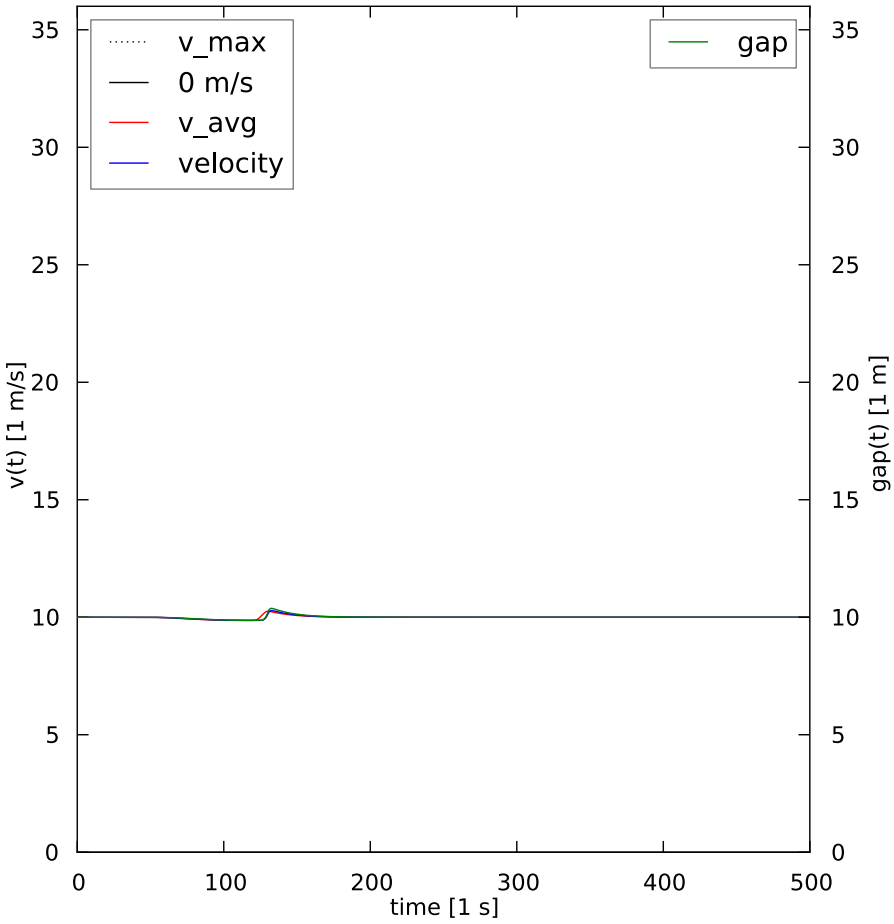


Figure 9.17.: 100^{th} vehicle with Jam-ADS.

Thus, the linearized Krauß model has the update rules

$$\begin{aligned} v(t+1) &= v_{\text{safe}}(t+1) \\ &= v_{\text{pred}}(t) + c(g(t) - v_{\text{pred}}(t)\tau) \\ &= cg(t) + (1 - c\tau)v_{\text{pred}}(t) \end{aligned} \tag{9.17}$$

and

$$\begin{aligned} g(t+1) &= g(t) + [v_{\text{pred}}(t) - v(t)]\,\Delta t \\ &= g(t) - c\Delta t\, g(t-1) + \Delta t\, v_{\text{pred}}(t) - (1 - c\tau)\Delta t\, v_{\text{pred}}(t-1) \quad . \end{aligned} \tag{9.18}$$

For simplification we assume t to be in units of Δt and all $v(t)$ and $g(t)$ appropriately adapted. This is opposed to the presentation of the Krauß model in Section 3.2.5. Note, that the Krauß model constrains the reaction time to $\tau \geq \Delta t$, thus c is constrained to $c < 1$ if we have a useful equilibrium velocity $v_0 > 0$.

We now use the index k as the vehicle number with $k = 0$ for the first (unperturbed) vehicle and $k = 1$ for the second (initially perturbed) vehicle and so forth. With this the update rules are a complete set of equations describing the whole system:

$$\begin{aligned} g_k(t+1) - g_k(t) + c\Delta t\, g_k(t-1) &= \Delta t\,[v_{k-1}(t) - (1 - c\tau)\, v_{k-1}(t-1)] \\ v_k(t+1) &= cg_k(t) + (1 - c\tau)\, v_{k-1}(t) \end{aligned} \tag{9.19}$$

with the boundary conditions:

$$\begin{aligned} g_0(t) &\equiv g_0 \\ v_0(t) &\equiv v_0 \end{aligned} \tag{9.20}$$

and the initial conditions:

$$\begin{aligned} g_1(0) &= g_0 - p\tau \\ v_1(0) &= v_0 - p \\ \forall k > 1 : g_k(0) &= g_0, v_k(0) = v_0 \quad . \end{aligned} \tag{9.21}$$

As before, p is the initial perturbation of the velocity of the second vehicle.

9.4.2. Time development of the linearized Krauß model

Figures 9.18 to 9.20 show the time development of velocities and gaps of the second, 35^{th}, and 100^{th} vehicle with the linearized Krauß model without Jam-ADS.

As can be seen in Figure 9.18, for the second vehicle there is only a marginal difference to the non-linear Krauß model. Data points above the red v_{avg} line are slightly lower and data points below are slightly higher, except for the given

initial perturbation. The same applies to the wave-packet-like oscillations of the 35^{th} vehicle (Figure 9.19). Only for the 100^{th} vehicle we observe a considerable difference, because it hits the zero and $v_{\max}$ limits with the original non-linear Krauß model. After linearization we no longer apply these limits, so linearization becomes less accurate when the oscillations grow outside of these limits.

Figure 9.21 shows the same functions as Figure 9.20 subtracted by their respective equilibrium values and plotted in double logarithmic scale. Due to the logarithmic scale the negative parts of the oscillations are missing. This plot shows that the oscillations never really cease, but their amplitudes drops more than exponentially. The oscillations in the plot vanish when the amplitude drops below the smallest positive numeric floating point values.

Additionally, the plot contains a black curve, which is a parabola in double logarithmic scales and comes close to an envelope function of the oscillations. Converting the double logarithmic parabola to linear values results in the function

$$h(t) = Ae^{-\alpha(\ln t - \ln T)^2} \quad . \tag{9.22}$$

T and A determine the horizontal and vertical position of the apex of the parabola and α determines its opening. By fitting these values manually we revealed that T, $\ln A$, and α grow linearly with the vehicle number. Hence, the time of the maximum amplitude of the oscillations is proportional to the vehicle number and the maximum amplitude grows exponentially with the vehicle number.

Interestingly, the time when the oscillations start and terminate is not affected at all by the linearization, even for the 100^{th} or any other vehicle we checked. Only the oscillations themselves look different when they reach the natural limits of zero and $v_{\max}$ in the non-linear case. This continues to apply if we change other configuration parameters.

Finally, we see in Figure 9.22 that Jam-ADS applied to the linearized Krauß model also has approximately the same effect as to the non-linearized Krauß model (Figure 9.16). The small remaining perturbation is more regular than in the non-linear case. Figure 9.23 shows the 100^{th} vehicle with Jam-ADS in double logarithmic scale with a hull parabola. With Jam-ADS the parabola does not serve as envelope so well, but we can at least see a massive drop of the oscillation amplitude and an earlier amplitude maximum, compared to the same parameter settings without Jam-ADS.

We now solve Equation (9.19) for the second, initially perturbed, vehicle. To do this we start with the equation

$$g_1(t) = A\gamma^t + B \quad . \tag{9.23}$$

This leads to two valid values for γ

$$\gamma_{\pm} = \frac{1}{2}\left(1 \pm \sqrt{1 - 4c\Delta t}\right) \quad . \tag{9.24}$$

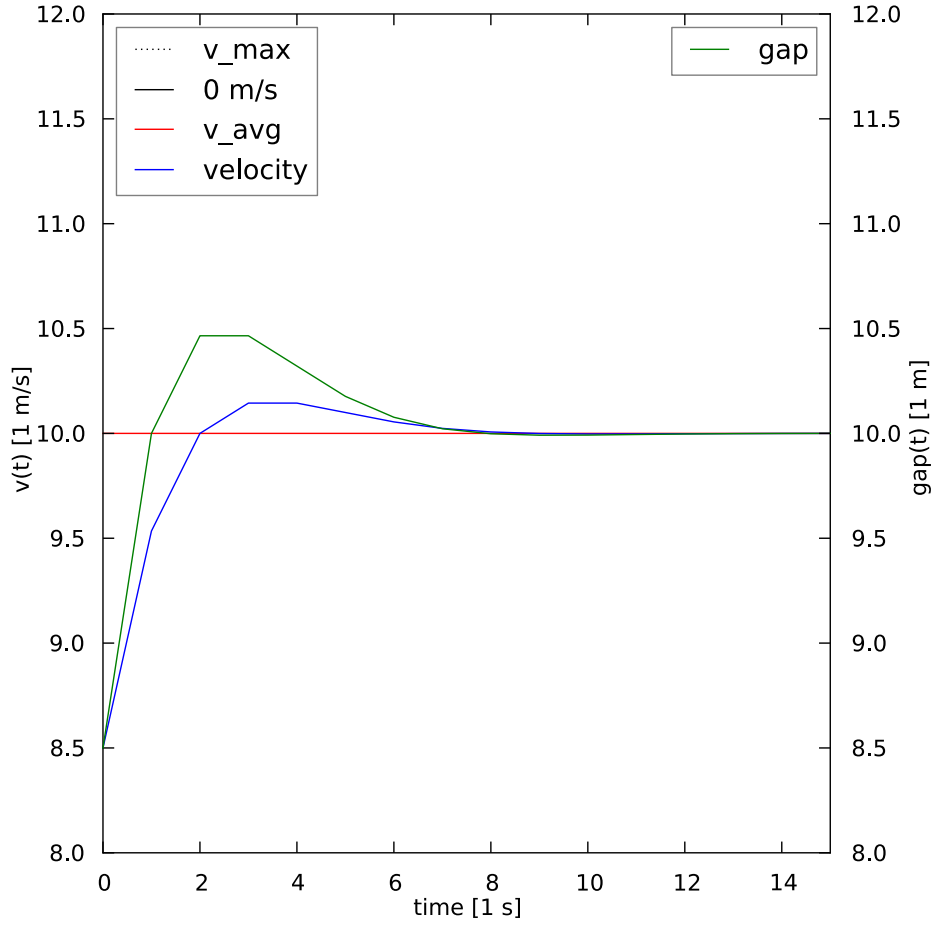


Figure 9.18.: Second (initially perturbed) vehicle with linearized Krauß model.

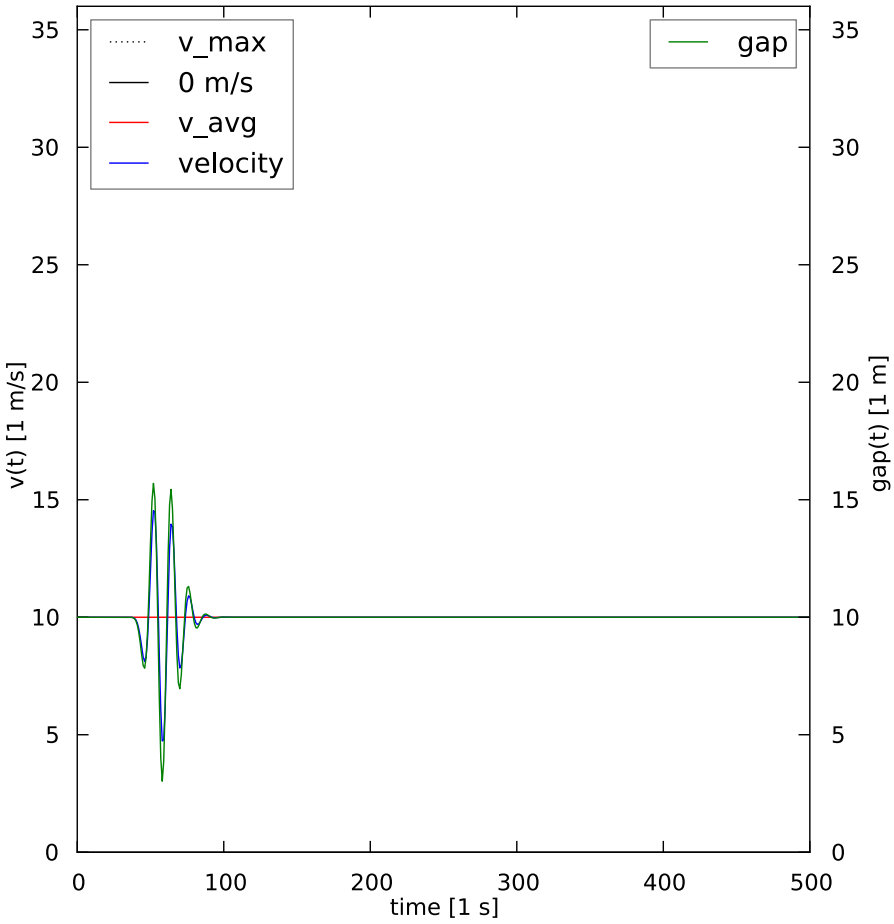


Figure 9.19.: 35^{th} vehicle with linearized Krauß model.

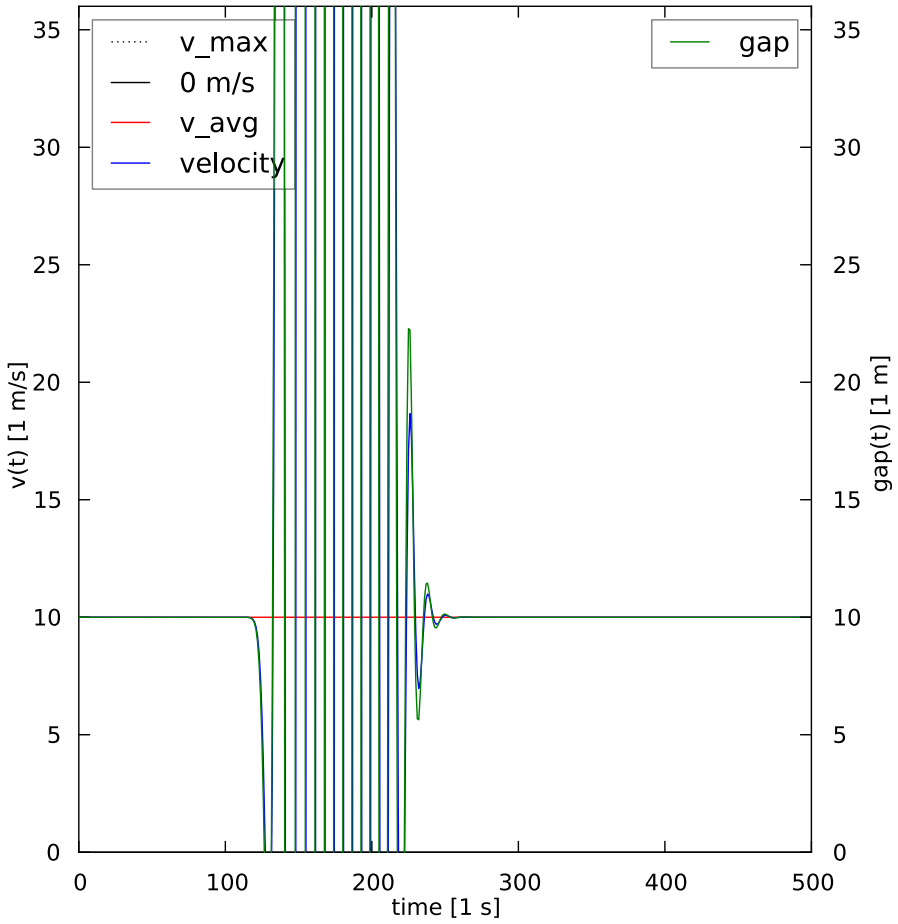


Figure 9.20.: 100^{th} vehicle with linearized Krauß model.

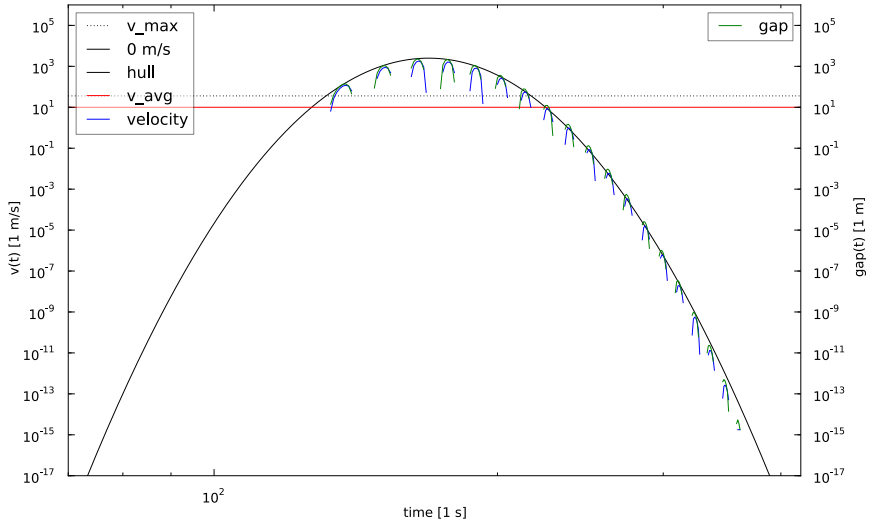


Figure 9.21.: 100^{th} vehicle with linearized Krauß model plotted in double logarithmic scale. The black hull curve is a parabola in this scale.

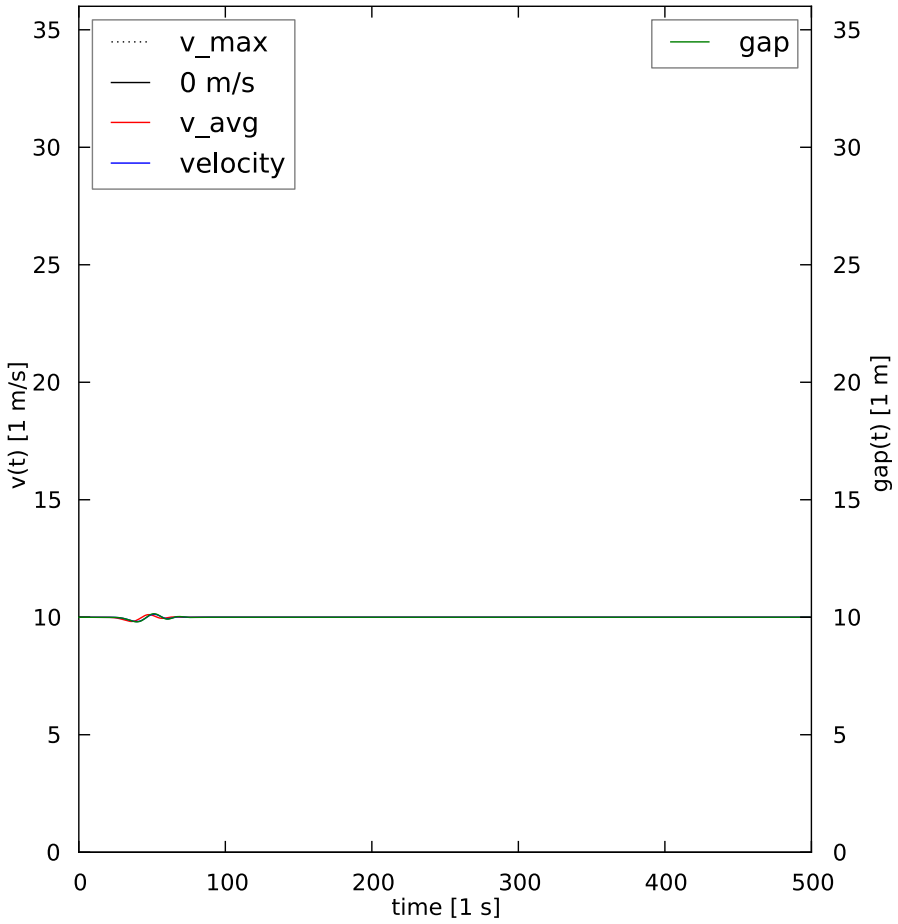


Figure 9.22.: 35^{th} vehicle with linearized Krauß model and Jam-ADS.

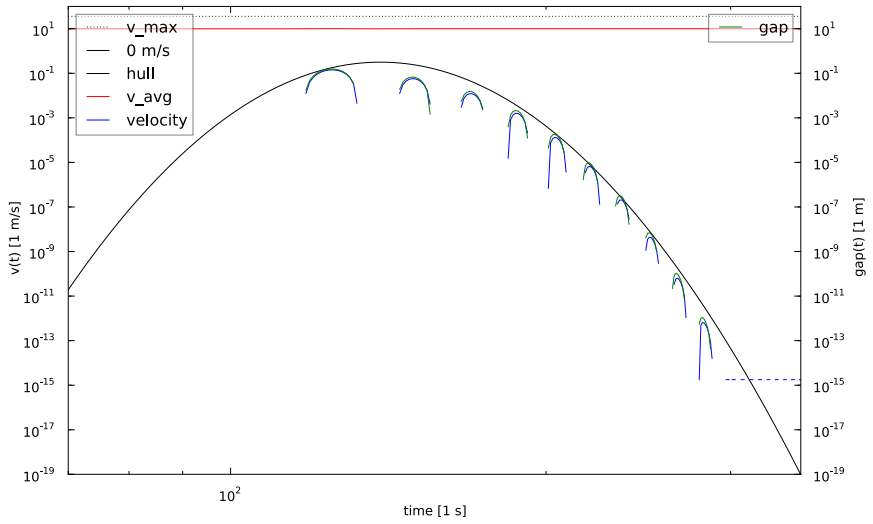


Figure 9.23.: 100^{th} vehicle with linearized Krauß model and Jam-ADS plotted in double logarithmic scale. The black hull curve is another parabola in this scale.

Thus, the final estimate is

$$g_1(t) = A_+\gamma_+^t + A_-\gamma_-^t + B \quad , \tag{9.25}$$

leading to the final solution

$$g_1(t) = g_0 + \frac{c\Delta t\, p\tau}{s}\left[\left(\frac{1+s}{2}\right)^{t-1} - \left(\frac{1-s}{2}\right)^{t-1}\right] \tag{9.26}$$

with

$$s := \sqrt{1 - 4c\,\Delta t} \quad . \tag{9.27}$$

Equation (9.26) even holds if s becomes purely imaginary, which is the case if $v_0 < b(4\Delta t - \tau)$. In that case $1 + s$ and $1 - s$ are complex conjugates of each other, so the square bracket is also purely imaginary. Together with the s in the denominator the result is real, again. With this in mind the expression can be transformed to a damped harmonic oscillation for these small enough v_0 as

$$g_1(t) = g_0 + \frac{2p\tau}{\sqrt{4c\,\Delta t - 1}}\sqrt{c\Delta t}^{t+1}\sin(\sigma(t-1)) \tag{9.28}$$

with a frequency σ defined by

$$\sin\sigma = \sqrt{1 - \frac{1}{4c\Delta t}} \quad . \tag{9.29}$$

Note, that the Krauß model is only valid if $\Delta t \leq \tau$, which is equivalent to $c\Delta t \leq 1$. Furthermore, from $v_0 > 0$ follows $c\Delta t < 1$. So, for non-vanishing equilibrium velocity v_0, $g_1(t)$ converges to g_0 with growing t.

Now we look at $g_1(t)$ under Jam-ADS. The second vehicle has only one vehicle ahead of itself, so the average velocity v_{avg} of the vehicles ahead is v_0. With this the recommended velocity (Equation (5.1)) becomes

$$v(t+1) = \lambda\, v_{\text{safe}}(t) + (1-\lambda)\, v_0 \quad , \tag{9.30}$$

which leads to the following new update rules, while the boundary and initial conditions (Equations (9.20) and (9.21)) are unchanged:

$$\begin{aligned} g_k(t+1) - g_k(t) + \lambda c\Delta t\, g_k(t-1) &= \Delta t\,[v_{k-1}(t) - (1-\lambda c\tau)\, v_{k-1}(t-1)] \\ v_k(t+1) &= \lambda c g_k(t) + (1-\lambda c\tau)\, v_{k-1}(t) \quad . \end{aligned} \tag{9.31}$$

Those are exactly the original update rules without Jam-ADS (Equation (9.19)), except that c is replaced by λc. Hence, the solution is

$$g_1(t) = g_0 + \frac{\lambda c\Delta t\, p\tau}{s_\lambda}\left[\left(\frac{1+s_\lambda}{2}\right)^{t-1} - \left(\frac{1-s_\lambda}{2}\right)^{t-1}\right] \tag{9.32}$$

with s_λ defined as

$$s_\lambda := \sqrt{1 - 4\lambda c\, \Delta t} \quad . \tag{9.33}$$

s_λ becomes purely imaginary for $v_0 < b(4\lambda\Delta t - \tau)$, which is smaller than the border to pure imaginary s without Jam-ADS, above. In this case we have

$$g_1(t) = g_0 + \frac{2p\tau}{\sqrt{4\lambda c\, \Delta t - 1}} \sqrt{\lambda c \Delta t}^{t+1} \sin(\sigma_\lambda (t-1)) \tag{9.34}$$

with an angular frequency σ_λ defined by:

$$\sin \sigma_\lambda = \sqrt{1 - \frac{1}{4\lambda c \Delta t}} \quad . \tag{9.35}$$

We see that the relaxation is quicker and the frequency becomes less than without Jam-ADS. So Jam-ADS damps the oscillations.

9.4.3. Linearized Krauß model in the frequency domain

Another approach to prove the effectiveness of Jam-ADS comes from control engineering. Control engineering provides the tools to evaluate the stability of controlled systems against external perturbations.[2] We apply some techniques of control engineering to describe the behavior of our chain of vehicles mathematically and show the effectiveness of Jam-ADS. Our approach is similar to the approach by Wilson [162, 163].

Let us consider a system as a black-box which transforms a time dependent input function $x : \mathbb{R}_+ \to \mathbb{R}$ (with $\mathbb{R}_+$ denoting the non-negative real numbers) into a time dependent output function $y : \mathbb{R}_+ \to \mathbb{R}$. If we assume linear and time-invariant behavior[3] of the transformation, it could be described by a linear differential equation in x and y with respect to time t.

The mathematical representation of the system's behavior can be simplified by applying the Laplace transform to the input and the output function. The Laplace transform $\mathcal{L}$ is defined as

$$F(s) = \mathcal{L}(f)(s) = \int_0^\infty f(t) e^{-st} dt \tag{9.36}$$

[2] Control engineering originated in the attempt to automate the task of helmsmen, who need to anticipate the ship's movement in advance to compensate the tardy steering of ships. This metaphorically relates to Jam-ADS, which helps the driver to anticipate the traffic situation in advance and properly react to it.

[3] Time-invariant means the behavior of the system does not depend on the absolute time. This is always the case for our system, like it is for most systems in the world.

for a function $f : \mathbb{R}_+ \to \mathbb{C}$, provided $f(t)e^{-st}$ is integrable.[4] The new independent parameter s is a complex number.[5] The Laplace transform is also known as transformation from time domain to frequency domain.

The Laplace transform converts a derivation in t into a multiplication with s, so the linear differential equation in time domain becomes a multiplication with a rational function in frequency domain, which is called the *transfer function* of the system:

$$Y(s) = G(s)X(s) = \frac{b_m s^m + \cdots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0} X(s) \quad , \tag{9.37}$$

where $Y := \mathcal{L}(y)$ and $X := \mathcal{L}(x)$ are the Laplace transforms of y and x. The summands of the numerator are the Laplace transformed derivations of x and the summands in the denominator are the Laplace transformed derivations of y of the differential equation in the time domain. We set $a_n = 1$ in the denominator, because other values are assumed to be factored out in the numerator and denominator.

Our system is discrete in time, so the description in frequency domain can be further simplified by applying the unilateral $\mathcal{Z}$-transform instead of the Laplace transform. Suppose we have a discrete function $\hat{f} : \mathbb{N}_0 \to \mathbb{C}$. Then the unilateral $\mathcal{Z}$-transform is defined by

$$F_z(z) = \mathcal{Z}(\hat{f})(z) = \sum_{n=0}^{\infty} \hat{f}(n) z^{-n} \quad . \tag{9.38}$$

This is the Laplace transform of a function f sampled at discrete points $n\Delta t$ in time (with δ as the Dirac δ-function):

$$f(t) = \sum_{n=0}^{\infty} \hat{f}(n)\delta(t - n\Delta t) \quad . \tag{9.39}$$

Together with the substitution

$$z = e^{s\Delta t} \tag{9.40}$$

we have the identity

$$\mathcal{L}(f)(s) = \mathcal{Z}(\hat{f})(z) \quad . \tag{9.41}$$

[4] In the literature the Laplace transform is often denoted by $\mathcal{L}\{f(t)\}$ instead of our $\mathcal{L}(f)$. We prefer the latter, because the Laplace transform is an operator mapping the whole function f on another function and not a function value $f(t)$ to another value. The Laplace transform is independent of the name of its parameter.

[5] If s is restricted to purely imaginary numbers ($s = i\omega$) the Laplace transform reduces to a Fourier transform provided f vanishes for negative values.

Just as the Laplace transform converts linear differential equations to rational functions, the $\mathcal{Z}$-transform converts linear difference equations to rational functions. Suppose we have a function $\hat{f}' : \mathbb{N}_0 \to \mathbb{C} : n \mapsto \hat{f}(n-1)$ (we extend $\hat{f}$ with $\hat{f}(-1) = 0$). Then insertion into Equation (9.38) yields

$$\mathcal{Z}(\hat{f}')(z) = \frac{1}{z}\mathcal{Z}(\hat{f})(z) \quad . \tag{9.42}$$

We now rearrange the update rules (Equation (9.19)), such that we eliminate the gap. From now on, we consider t as an integer multiple of Δt again. Our initial update rules were

$$v_k(t+1) = cg_k(t) + (1 - c\tau)\, v_{k-1}(t) \tag{9.43}$$
$$g_k(t+1) = g_k(t) + \Delta t\, [v_{k-1}(t) - v_k(t)] \quad . \tag{9.44}$$

To join these equations into one equation we start with the expression $v_k(t+1) - v_k(t)$ and insert Equation (9.43) for both summands (with a shifted t in case of $v_k(t)$):

$$v_k(t) - v_k(t-1) = cg_k(t) + (1-c\tau)\, v_{k-1}(t) - cg_k(t-1) + (1-c\tau)\, v_{k-1}(t-1). \tag{9.45}$$

Now, we insert Equation (9.44) for $g_k(t)$ with shifted t and reorder the summands to have index k on the left and index $k-1$ on the right:

$$v_k(t) - v_k(t-1) + c\Delta t\, v_k(t-2) = (1-c\tau)\, v_{k-1}(t-1) - (1 - c(\tau + \Delta t))\, v_{k-1}(t-2)\,. \tag{9.46}$$

This is an equation describing the transition of the velocity from vehicle $k-1$ to vehicle k.

We are interested in the oscillations around the equilibrium velocity v_0 and equilibrium gap $g_0 = \tau v_0$, thus we need to shift v_k and g_k such that they denote the difference to the equilibrium velocity

$$\hat{g}_k(t) := g_k(t) - g_0 \tag{9.47}$$
$$\hat{v}_k(t) := v_k(t) - v_0 \tag{9.48}$$

and substitute these in Equation (9.46). This results in the same equations for $\hat{v}_k$ and $\hat{g}_k$ as those for v_k and g_k, because the equilibrium cancels on both sides. Thus, we do not have to consider the equilibrium of the oscillations in the following thoughts. To keep things simple we stick with the symbols v_k and g_k.

Applying the $\mathcal{Z}$-transform on both sides of Equation (9.46) with $V_k = \mathcal{Z}(v_k)$ yields:

$$\left[1 - z^{-1} + c\Delta t z^{-2}\right] V_k(z) = \left[(1-c\tau)z^{-1} - (1 - c(\tau + \Delta t))z^{-2}\right] V_{k-1}(z) \quad . \tag{9.49}$$

So, in frequency domain we have the following transfer function G transferring the behavior of the vehicle $k-1$ to the behavior of vehicle k:

$$\begin{aligned} G(z) &= \frac{V_k(z)}{V_{k-1}(z)} \\ &= \frac{(1-c\tau)z^{-1} - (1-c(\tau+\Delta t))z^{-2}}{1 - z^{-1} + c\Delta t z^{-2}} \\ &= \frac{(1-c\tau)z - (1-c(\tau+\Delta t))}{z^2 - z + c\Delta t} \quad . \end{aligned} \tag{9.50}$$

This transfer function has two poles we call z_+ and z_-:

$$z_\pm = \frac{1}{2}\left(1 \pm \sqrt{1-4c\Delta t}\right) \quad . \tag{9.51}$$

c is always positive, hence the square root is either imaginary or non-negative real and less than one. Thus, the poles have always positive real parts.

The transfer function has only a single zero z_0 at

$$z_0 = 1 - \frac{c\Delta t}{1-c\tau} \quad . \tag{9.52}$$

Knowing the zeros and poles and with partial fraction decomposition G can be expressed as

$$\begin{aligned} G(z) &= (1-c\tau)\frac{z-z_0}{(z-z_+)(z-z_-)} \\ &= \frac{1-c\tau}{\sqrt{1-4c\Delta t}}(z-z_0)\left[\frac{1}{z-z_+} - \frac{1}{z-z_-}\right] \quad . \end{aligned} \tag{9.53}$$

The reaction of the system to harmonic oscillations of a given circular frequency ω is of special interest, because linear systems amplify or damp independently harmonic oscillations of different frequency. Harmonic oscillations correspond to input functions $x(t) = e^{st} = e^{i\omega t}$, which means we have a pure imaginary $s = i\omega$. In case of the $\mathcal{Z}$-transform according to the substitution in Equation (9.40) this corresponds to $z = e^{s\Delta t} = e^{i\omega\Delta t}$.

We can display this graphically by using Bode diagrams (Figures 9.24 to 9.28). Bode diagrams consist of two sub-diagrams with the same x-axis displaying ω. The y-axis of the top sub-diagram indicates the absolute value of the transfer function and the y-axis of the bottom sub-diagram indicates its phase angle.[6] The phase angle is periodic, so top and bottom edges of the phase sub-diagram are identical.

[6] Our Bode diagrams show all angles in angular degrees instead of radians for easier identification of special angles.

Thus, for example, the vertical line at 180° in the phase sub-diagram of Figure 9.24 is artificial.

In case of the $\mathcal{Z}$-transform the substitution $z = e^{i\omega\Delta t}$ is periodic in ω. Adding an integer multiple of $2\pi/\Delta t$ to ω does not change the value of z. All of our presented Bode diagrams cover one period of ω on the x-axes, so left and right edges are always identical like the top and bottom edges of the phase-angle sub-diagrams. Additionally, around integer multiples of $\pi/\Delta t$ the absolute value of the transfer function is symmetric and the phase angle anti-symmetric. Appendix C gives a proof for this.

Computing velocities and gaps in time steps Δt corresponds to sampling with a frequency $1/\Delta t$. According to the Nyquist-Shannon sample theorem oscillations with frequencies of half or more of the sample frequency cannot be reproduced, because there is always another oscillation with a lower frequency, which would produce the same sample points.[7] An angular frequency $\omega \geq \pi/\Delta t$ is a so-called alias of the angular frequency $2\pi/\Delta t - \omega$, which conforms to the symmetry around $\pi/\Delta t$.

The question is now whether there are oscillation frequencies ω that become amplified from one vehicle to the next, such that the amplitude grows more and more along the chain of vehicles. This corresponds to absolute values of the transfer function G larger than 1. Those are the points in the upper sub-diagrams of the Bode diagrams that are above the $10^0 = 1$ line (denoted as blue dotted lines in our Bode diagrams).

Figure 9.24 shows the Bode diagram of the transfer function $G(z(\omega)) = G(e^{i\omega\Delta t})$ with the configuration according to Table 9.1. These settings make $c \approx 0.310$. The left maximum is located at $\omega = 28.1°$ and becomes aliased at $\omega = 360° - 28.1° = 331.9°$. This corresponds to a period length of $360°/(28.1°/\Delta t) = 12.81\,\mathrm{s}$, which is the period length of the corresponding linear oscillations, seen above, within our accuracy of 1 s. The absolute values of the maxima are 1.103.

Figure 9.25 shows the transfer function $G^{100}(z(\omega)) = G^{100}(e^{i\omega\Delta t})$. The 100^{th} power is the transfer function over 100 vehicles along the vehicle chain. The maxima are at the same ω, but have an absolute value of 18343.9.[8] We see that absolute values above $10^0 = 1$ become amplified while absolute values below become damped, as expected.

Increasing the power of G—that is moving along the chain of vehicles—always keeps the locations of the maxima, but their absolute values increase. Changing the equilibrium velocity v_0 always reproduces the oscillation period of the "wave packet" in time domain as maximum of the Bode diagram.

[7] This is why audio signals are usually sampled with more than 40 kHz to capture all harmonics up to the limit of 20 kHz considered as limit of human hearing. Technical reasons could require higher audio sample frequencies.

[8] $1.103^{100} = 18094.7$, but if we consider enough digits the maximum of G^{100} equals the 100^{th} power of the maximum of G.

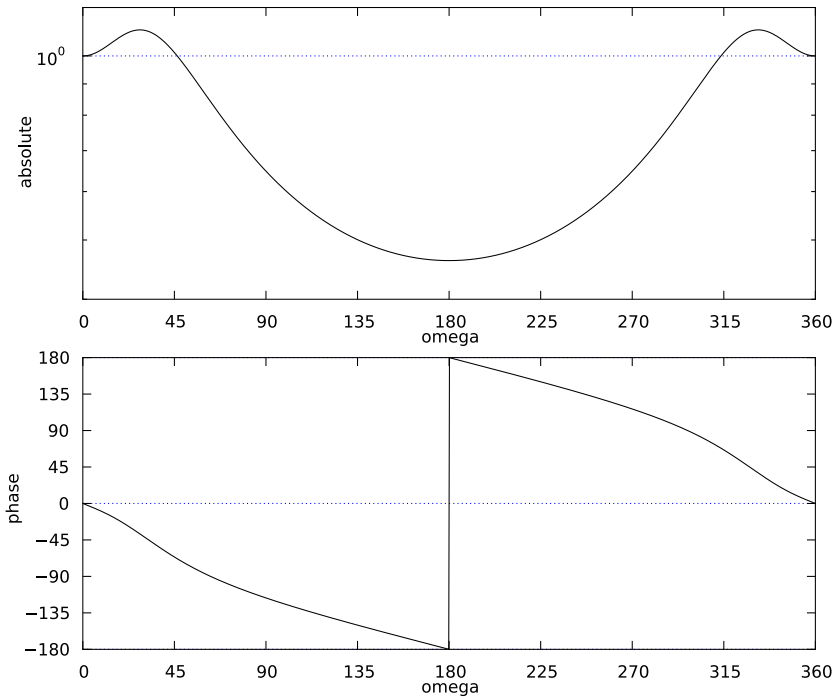


Figure 9.24.: Bode diagram of transfer function $G(e^{i\omega\Delta t})$ with $\Delta t = 1\,\mathrm{s}$ of linearized Krauß model.

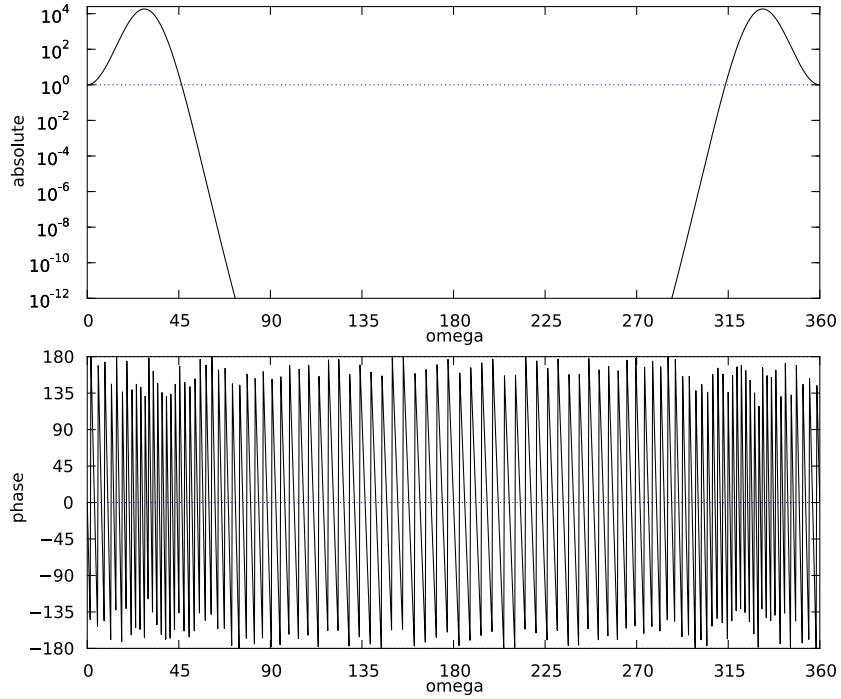


Figure 9.25.: Bode diagram of transfer function $G^{100}(e^{i\omega\Delta t})$ with $\Delta t = 1\,\mathrm{s}$ of linearized Krauß model corresponding to oscillation transfer over 100 vehicles.

Setting $v_0 = 0$, corresponding to $c = 1/\tau = 1/\Delta t$, yields

$$z_\pm = \frac{1}{2}\left(1 \pm i\sqrt{3}\right) = e^{\pm i\frac{\pi}{3}\Delta t} \quad . \tag{9.54}$$

Thus, in this case we have infinite maxima at $\omega = 60°$ and—because of symmetry—$\omega = 300°$.

The phase angle of the Bode diagrams is of no interest for us. However, we briefly explain why it looks so noisy in Figure 9.25. As already mentioned, the phase sub-diagram is periodic in the y-axis. Computing the k^{th} power of G multiplies the phase angle with k. Thus, because of the periodicity we receive an overflow whenever the phase angle of G passes an integer multiple of $180°/k$, which is quite often for $k = 100$.

It might seem contradictory that we found oscillation frequencies that become always amplified along the chain of vehicles, although our previous results on a single perturbation in time domain showed that all oscillations are damped to negligible fluctuations of the equilibrium. However, the difference is that the Bode diagrams show the reaction to a permanent harmonic oscillation of the predecessor while the previous results show the reaction to a single perturbation. Thus, we have different reactions to different perturbations.

9.4.4. Jam-ADS with linearized Krauß model

We now insert Jam-ADS into the transfer function G to see its effect. Without Jam-ADS the update rule (Equation (9.46)) was

$$v_k(t) = \alpha_1 v_k(t-1) + \alpha_2 v_k(t-2) + \beta_1 v_{k-1}(t-1) + \beta_2 v_{k-1}(t-2) \tag{9.55}$$

with

$$\begin{aligned} \alpha_1 &= 1 \\ \alpha_2 &= -c\Delta t \\ \beta_1 &= 1 - c\tau \\ \beta_2 &= -(1 - c(\tau + \Delta t)) \quad . \end{aligned} \tag{9.56}$$

Now we apply Jam-ADS by building the convex combination with the average velocity of the N vehicles ahead:

$$\begin{aligned} v_k(t) =& \lambda\Big[\alpha_1 v_k(t-1) + \alpha_2 v_k(t-2) + \beta_1 v_{k-1}(t-1) + \beta_2 v_{k-1}(t-2)\Big] \\ &+ (1-\lambda)\frac{1}{N}\sum_{\ell=1}^{N} v_{k-\ell}(t-1) \quad . \end{aligned} \tag{9.57}$$

Applying the $\mathcal{Z}$-transform on both sides and multiplying with z^2 yields in frequency domain

$$\begin{aligned} z^2 V_k(z) =& \lambda\Big[\alpha_1 z V_k(z) + \alpha_2 V_k(z) + \beta_1 z V_{k-1}(z) + \beta_2 V_{k-1}(z)\Big] \\ &+ (1-\lambda) z \frac{1}{N} \sum_{\ell=1}^{N} V_{k-\ell}(z) \quad . \end{aligned} \tag{9.58}$$

Ordering the summands by their vehicle indices we get

$$\begin{aligned} \left(z^2 - \lambda\alpha_1 z - \lambda\alpha_2\right) V_k(z) =& \left(\lambda\beta_1 z + \lambda\beta_2 + \frac{1-\lambda}{N} z\right) V_{k-1}(z) \\ &+ (1-\lambda) z \frac{1}{N} \sum_{\ell=2}^{N} V_{k-\ell}(z) \quad . \end{aligned} \tag{9.59}$$

This allows us to define two transfer functions

$$P_N(z) := \frac{(\lambda\beta_1 + \frac{1-\lambda}{N})z + \lambda\beta_2}{z^2 - \lambda\alpha_1 z - \lambda\alpha_2} = \frac{(\lambda(1 - c\tau) + \frac{1-\lambda}{N})z - \lambda(1 - c(\tau + \Delta t))}{z^2 - \lambda z + \lambda c \Delta t} \tag{9.60}$$

$$Q(z) := \frac{(1-\lambda)z}{z^2 - \lambda\alpha_1 z - \lambda\alpha_2} = \frac{(1-\lambda)z}{z^2 - \lambda z + \lambda c \Delta t} \quad . \tag{9.61}$$

With these transfer functions we can express the $\mathcal{Z}$-transform of the velocity of vehicle k by the $\mathcal{Z}$-transform of the velocity of the direct predecessor $k-1$ and the $\mathcal{Z}$-transform of the velocities of the $N-1$ vehicles further ahead:

$$V_k(z) = P_N(z) V_{k-1}(z) + Q(z) \frac{1}{N} \sum_{\ell=2}^{N} V_{k-\ell}(z) \quad . \tag{9.62}$$

It is correct to divide the sum by N although it contains only $N-1$ summands, because the contribution of V_{k-1} to the average is already contained in $P_N(z)$.

Figure 9.26 contains the Bode diagram of P_{20} ($N = 20$) and Figure 9.27 the Bode diagram of Q for $\lambda = 0.6$ with the same parameter values as before ($\tau = \Delta t = 1\,\mathrm{s}$, $v_0 = 10\,\mathrm{m/s}$, $b = 4.5\,\mathrm{m/s^2}$ resulting in $c \approx 0.310$). We can see the absolute values of both functions are completely below the critical $10^0 = 1$; P_{20} has a maximum of 0.533 and Q a maximum of 0.684.

It is possible to compute a joined transfer function from the first vehicle V_1 to

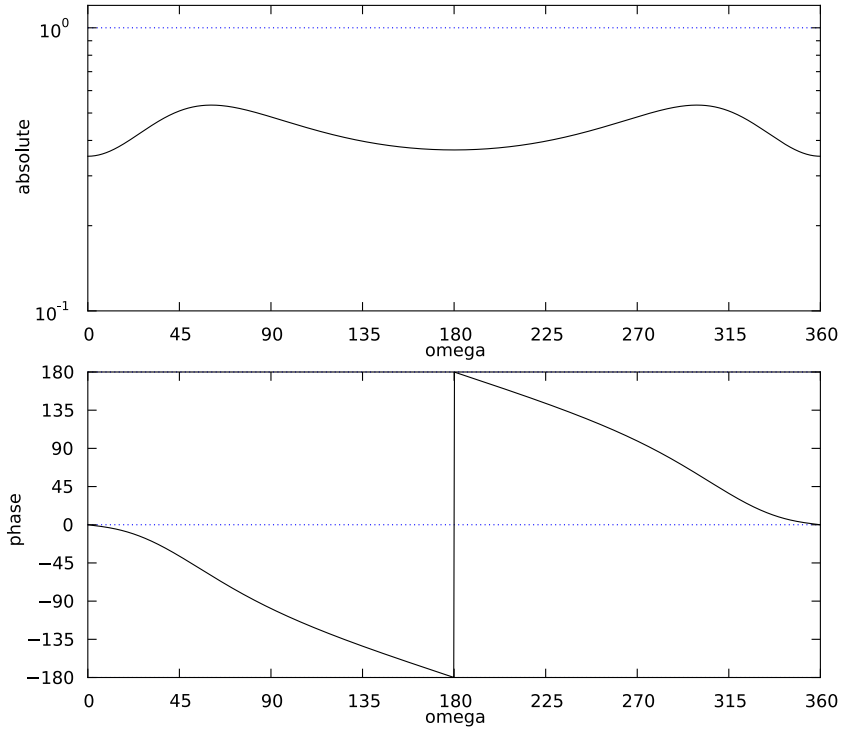


Figure 9.26.: Bode diagram of transfer function P_{20} of linearized Krauß model with $\lambda = 0.6$ and $N = 20$.

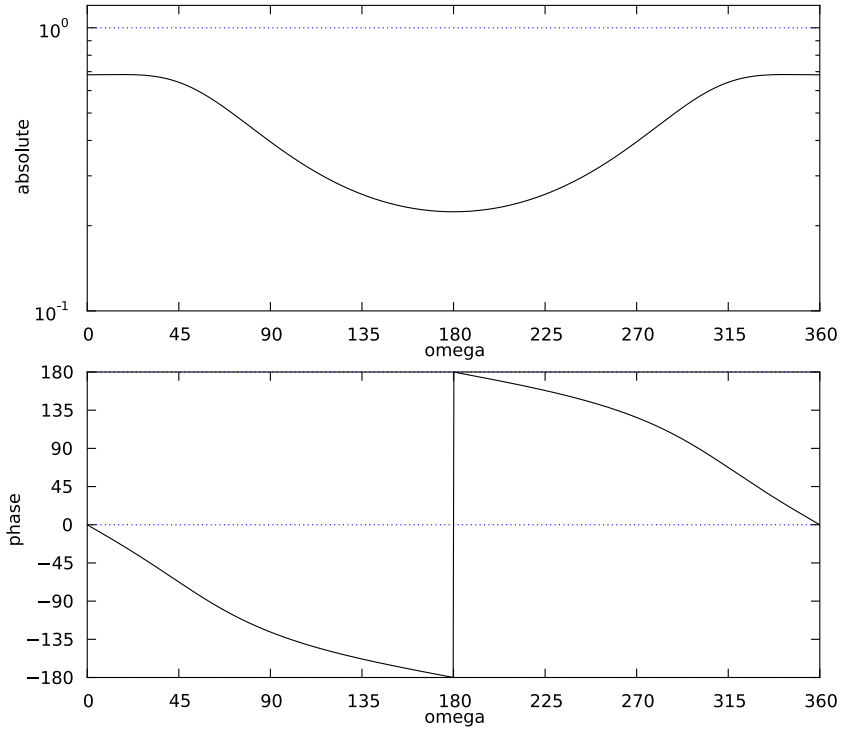


Figure 9.27.: Bode diagram of transfer function Q of linearized Krauß model with $\lambda = 0.6$ (it does not depend on N).

the k$^{\text{th}}$ vehicle V_k by recursively applying Equation (9.62) to V_1 (with $n \in \mathbb{N}$):

$$\begin{aligned}
V_1(z) &= \mathbb{1}V_1(z) && = G_1(z)V_1(z)\\
V_2(z) &= P_1(z)V_1(z) && = G_2(z)V_1(z)\\
V_3(z) &= P_2(z)V_2(z) + \frac{1}{2}Q(z)V_1(z)\\
&= \left[P_2(z)P_1(z) + \frac{1}{2}Q(z)\right]V_1(z) && = G_3(z)V_1(z)\\
&\vdots\\
V_{N+n}(z) &= P_N(z)V_N(z) + \frac{1}{N}Q(z)\sum_{\ell=2}^{N} V_{N-\ell}(z) && = G_{N+n}(z)V_1(z) \quad .
\end{aligned}$$

Note that for computing V_1 through V_N we applied P_1 through P_N in the recursion and divided $Q(z)$ by 2 to N for $k = 2$ to $k = N$, respectively. From V_{N+n} on we stay with P_N and the divisor N, because in the first N steps we have fewer than N vehicles to average over.

Unfortunately, the resulting joined transfer function is not stable. So, we modify the strategy to apply always P_N and always divide by N:

$$\begin{aligned}
V_1(z) &= \mathbb{1}V_1(z) && = G_1(z)V_1(z)\\
V_2(z) &= P_N(z)V_1(z) && = G_2(z)V_1(z)\\
V_3(z) &= P_N(z)V_2(z) + \frac{1}{N}Q(z)V_1(z)\\
&= \left[P_N^2(z) + \frac{1}{N}Q(z)\right]V_1(z) && = G_3(z)V_1(z)\\
&\vdots\\
V_{N+n}(z) &= P_N(z)V_N(z) + \frac{1}{N}Q(z)\sum_{\ell=2}^{N} V_{N-\ell}(z) && = G_{N+n}(z)V_1(z) \quad .
\end{aligned}$$

We can do this because it is a variant of Jam-ADS, that we are free to select.

Figure 9.28 shows the result of this recursion for the joined transfer function G_{100} from V_1 to V_{100}. Its maxima have a value of 0.267.

Changing N shows that a larger N makes the maximum value of P_N and of the joined transfer function smaller. From Equation (9.60) we can see this effect on P_N and as the joined transfer function is a sum over positive products of P_N and Q the joined transfer function must exhibit the same behavior. In our example of the 100^{th} vehicle N must be at least 13 to keep the entire joined transfer function below $10^0 = 1$ if the other parameters remain unchanged. For larger vehicle numbers N must be raised as well but less than proportional to the vehicle number.

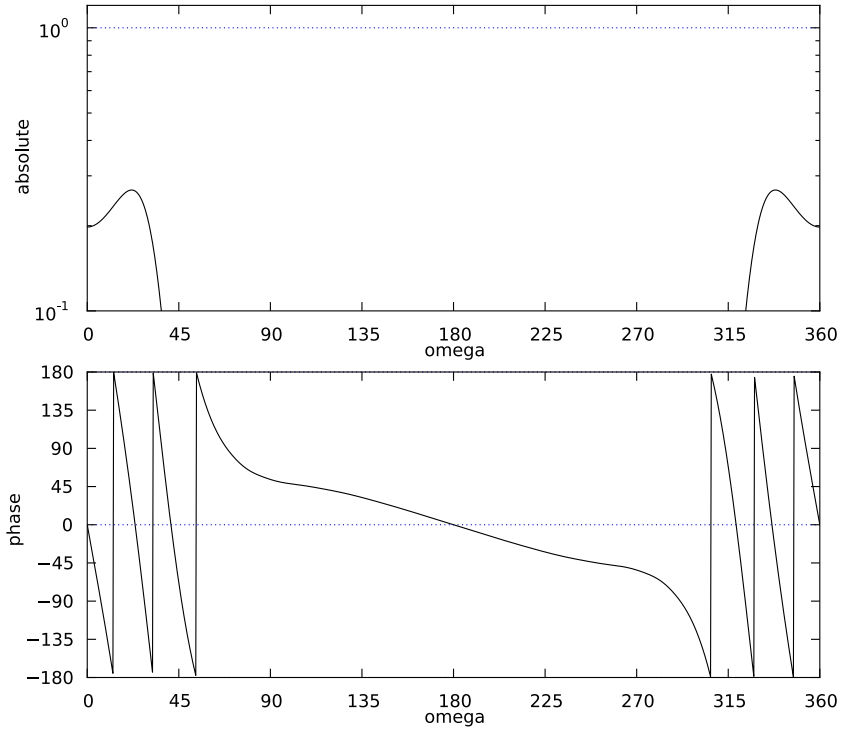


Figure 9.28.: Bode diagram of joined transfer function G_{100} of linearized Krauß model from first to 100^{th} vehicle with $\lambda = 0.6$ and $N = 20$.

A smaller λ makes the absolute value of P smaller, but increases the absolute value of Q. The joined transfer function decreases with decreasing λ, but if $\lambda < 0.9$ the decrease becomes small for vehicles further down the chain. In other words, the difference between $\lambda = 0.75$ and $\lambda = 0.6$ is negligible, for example.

For larger equilibrium velocity v_0, corresponding to a smaller c, N needs to be larger (or λ smaller) to reach the stable condition. However, even for $v_0 = 80\,\mathrm{m/s} = 288\,\mathrm{km/h}$ averaging over $N = 23$ vehicles is sufficient. For very small equilibrium velocities below $v_0 = 1\,\mathrm{m/s}$ the joined transfer function has another maximum that cannot be removed by changing λ or N. Thus, the strategy should be turned off for very small gaps and velocities.

Trying different parameter settings has shown that for nearly all conditions Jam-ADS produces a joined transfer function below the critical $10^0 = 1$, which means that all perturbation frequencies become damped along the chain of vehicles, but without Jam-ADS there are always perturbation frequencies that become amplified. Thus, Jam-ADS is capable of making a chain of vehicles stable in terms of congestion-causing perturbations.

10. Urban Traffic

Up to now we investigated the improvement of flow on highways using car-to-car communication. We also thought about applying local communication to improve urban traffic flow. The first step from the highway setting towards urban traffic is to add traffic lights and assume communication between vehicles and traffic lights additional to the communication among the equipped vehicles.

There are basically two approaches to utilize the information exchange between vehicles and traffic lights. By receiving position and velocity information from the vehicles, traffic lights may know the traffic demand better and adopt their schedule accordingly. Currently, traffic-actuated traffic lights are usually coupled with a single induction loop positioned at a certain distance. Wireless communication could provide more detailed information about equipped vehicles while the detector loop provides information about the remaining unequipped vehicles. Utilized in the other direction, vehicles could know the timing of the upcoming traffic light phases in advance and adapt their approach velocity accordingly. Strategies to adapt the velocity need to incorporate the predecessing vehicles up to the next traffic light as well, which requires additional car-to-car communication.

Papageorgiou et al. [112] give an overview over various traffic-light control strategies classified into fixed-time and traffic-responsive (usually actuated by induction loops) strategies and also classified into isolated (only one intersection) and coordinated strategies. Newell [110] dynamically adapts the length of the green phase to minimize the overall delay of all vehicles at an intersection. Lo [100] has extended the cell-transmission model (CTM), which is a cellular automaton based on the Lighthill-Whitham-Richards model (Section 3.1), to accommodate traffic lights, and has applied integer programming to the CTM to optimize traffic light schedules. Appl and Sollacher [1] select the best fixed-time scheduling plan from a given list of plans for each time of the day by observing traffic demand for some days. Heung, Ho, and Fung [58] minimize delays by a fuzzy-logic controller whose parameters are dynamically adapted by genetic algorithms. In addition, they apply dynamic programming to coordinate neighboring intersections to further reduce the delay and the number of stops.

All papers mentioned so far suggest strategies without making use of car-to-infrastructure communication. Wenjie et al. [158] apply C2I communication to estimate the sizes of the vehicle queues at an intersection to dynamically adapt the traffic-light phase lengths. Similarly, Gradinescu et al. [44] adapt the cycle

length of an intersection[1] by applying well-known optimization algorithms.

Less research has been done regarding the other direction of information flow from the traffic lights to the vehicles. However, most of the research projects about intelligent transportation systems presented in Section 4.7 investigate strategies to improve how vehicles approach traffic lights considering information about the next phases.

Our goal is to find a strategy similar to Jam-ADS incorporating the knowledge of traffic light schedules to improve the flow over a number of successive traffic lights. In this chapter we present some intermediate results.

10.1. Traffic breakdown example

As testbed for different ideas we extended CircSim to simulate traffic lights, as described in Section 6.3.1. As a first experiment we set a maximum velocity of $v_{\max} = 50\,\mathrm{km/h}$ and configured several evenly distributed traffic lights, such that their schedules support flow at $v_{\max}$, which is known as *progressive signal system* or *green wave*.

Without random deceleration the vehicles become organized in platoons, which move along the circular road as if no traffic lights are present (Figure 10.1 shows a snapshot of the watch plot). We see that the average velocity equals exactly $v_{\max} = 50\,\mathrm{km/h}$ and there is no fuel consumption caused by acceleration, although all traffic lights are red at the instant of the snapshot. No vehicle has to slow down, because the platoons fit completely into the green phases.

Figure 10.2 shows what happens if random deceleration is turned on with $\varepsilon = 1$ (see Section 3.2.5 for definition of ε). The average velocity drops dramatically and the fuel consumption increases massively because of the acceleration consumption. The less air- and rolling-resistance consumption can not compensate for that. Above all, there is some idle consumption. Furthermore, we see that the average velocity during the green phase (Figure 10.2a) is less than the average velocity during the red phase (Figure 10.2b), because the green wave is out of sync with the velocity of the platoons.

Turning off the random deceleration again quickly restores the original situation shown in Figure 10.1. If a platoon is too long to fit into a green phase, the next red phase shifts the last vehicles of the platoon to the next upstream one until all platoons completely fit into the green phases.

Two effects of the random deceleration cause the breakdown. First, the random deceleration increases the equilibrium gap between successive vehicles, and second, it decreases the expected velocity as explained in Section 3.2.5. If we reduce the velocity of the green wave to the expected velocity with random deceleration and

[1] *Cycle length* is the total duration of all traffic-light phases at an intersection after which the phases are usually repeated.

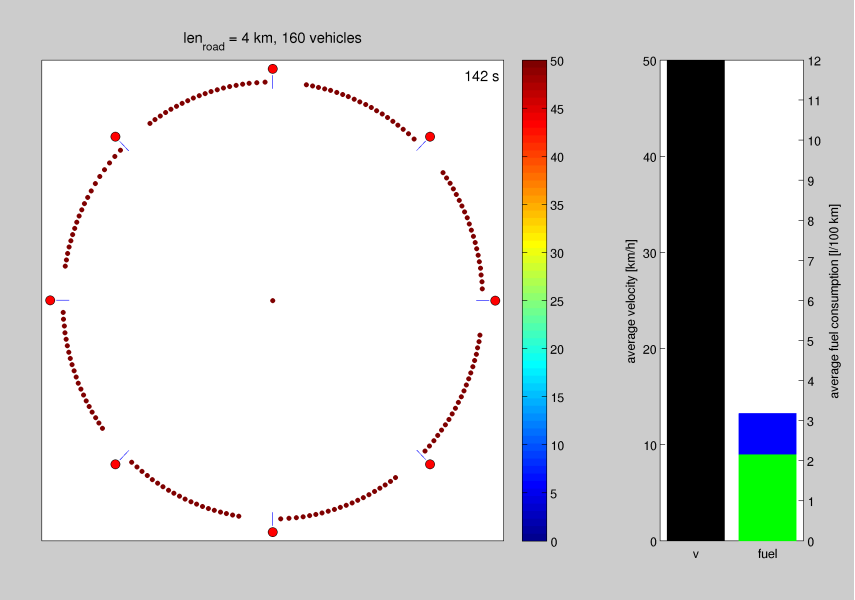


Figure 10.1.: Green-wave flow without random deceleration.

additionally reduce the number of vehicles to compensate for the longer platoons[2] then we obtain the perfect green wave behavior of Figure 10.1 with random deceleration and the breakdown behavior of Figure 10.2 without random deceleration.

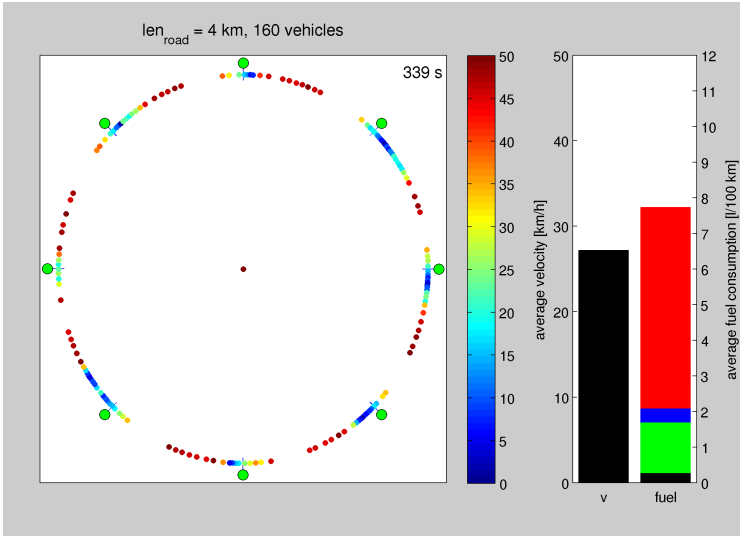
However, this example shows that slight changes of the parameters can cause huge changes of the traffic situation. We expect that proper adaptations of the vehicles behavior can mitigate bad traffic situations like in this example.

10.2. Idea of a traffic-light approach strategy

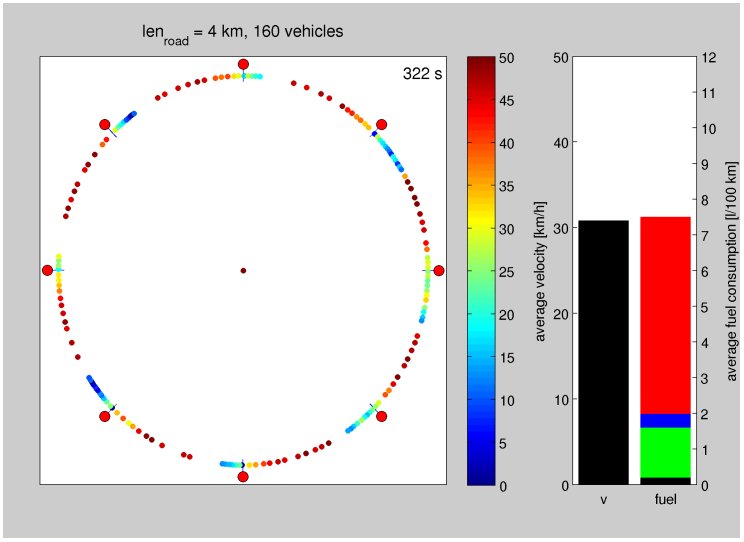
We assume that without any strategy all vehicles move as fast as safely and legally possible, thus a traffic light approach strategy can only make vehicles move slower.

A single vehicle is not only delayed by stopping at a red light, but also by moving slower than the maximum velocity during the acceleration stage after the light becomes green again. Thus, for a single vehicle approaching a red traffic light—or a traffic light which would turn to red before the vehicle may reach it—the best strategy would be to slow down such that the light is reached with

[2]The limited initial population length of CircSim described in Section 6.3.1 helped to measure the equilibrium length of platoons with random deceleration.



(a) Green phase



(b) Red phase

Figure 10.2.: Green-wave flow with random deceleration.

the highest possible velocity at the moment it turns green again, because that would minimize the time loss during the acceleration stage. This implies that an approaching vehicle might massively decelerate or even stop at a distance to the traffic light and reaccelerate during the end of the red phase such that the light is passed with $v_{\max}$ when it turns green. However, such a strategy is not the best in terms of energy consumption compared to strategies which do not reaccelerate before reaching the light, because these require more acceleration in total, which costs more fuel. In addition, stopping at a distance to the red light could have bad effects to the upstream traffic and might even be illegal. Hence, we propose as single vehicle strategy to adapt the deceleration when approaching a red light such that it is reached with the highest possible remaining velocity at the moment it turns green.

Multiple vehicles make the situation more complicated. In this case the strategy has to consider all vehicles between ours and the traffic light as well. These vehicles need their share of the next green phase before our vehicle can pass the light. They might even need more than one green phase such that our vehicle's time loss comprises several red and intermediate green phases. Thus, the presence of other vehicles shifts the earliest moment our vehicle could pass the traffic light to a later time. This could be incorporated into the single vehicle strategy above by inserting the estimated earliest passing time. However, we have to also consider that the other vehicles may occupy the space we need for our strategy.

An idea to simplify the strategy for multiple vehicles is to apply it only to the leaders of platoons, as the other vehicles of a platoon are forced to slow down to the velocities of their leaders and follow them as closely as possible. If not all vehicles of a platoon are estimated to pass during the designated green phase of the leader, the first vehicle which is estimated not to pass is declared as another platoon leader and as such slows down to arrive at the start of the next green phase. On multiple-lane roads there has to be one platoon leader per lane. When platoon leaders switch lanes the leadership has to be reassigned. Having only the assigned platoon leaders follow the strategy has the additional advantage that not all vehicles have to be equipped with wireless communication, although a higher equipment rate helps to find more appropriate platoon leaders and to better evaluate the current traffic situation.

To compute the maximum number of vehicles of a platoon expected to pass during a single green phase, the traffic light dynamically determines the average maximum passing rate of vehicles during green phases and disseminates this information to the vehicles. To find the maximum passing rate it is not sufficient to count the vehicles which passed during the last green phase, because those might be fewer than maximally possible. Instead, the traffic light should also consider gaps larger then necessary and compute how many additional vehicles would fit into the gaps on average.

11. Conclusion and Outlook

11.1. Conclusion

We proposed the driver assistance strategy Jam-ADS to improve the flow of traffic based on C2C communication, tested it with extensive simulations, and analyzed it mathematically. In Section 5.3.1 we drew up a list of objectives to which Jam-ADS should conform. In the following we summarize if and how these objectives are achieved:

Objective 1 *It should depend entirely on local communication among the vehicles:*
This objective is achieved by design of Jam-ADS. All suggested Jam-ADS sub-strategies only need positions and velocities of the vehicles ahead. With the AutoCast protocol, developed by the AutoNomos project, this information can be gained by pure local communication.

Objective 2 *It should prevent traffic jams and congestion:*
Our simulations confirmed that Jam-ADS prevents congestion if properly configured. In addition, we showed analytically for a simplified case that Jam-ADS damps all harmonic frequencies of perturbations.

Objective 3 *It should resolve existing traffic jams:*
Our simulations of the closed system showed the ability of Jam-ADS to resolve existing traffic jams in case of both single and multiple lanes.

Objective 4 *It should reduce fuel consumption:*
In our simulations of the single-lane closed system we saw a considerable reduction of fuel consumption. Jam-ADS in the multiple-lane closed system and continuous Jam-ADS in the open system exhibited a slight reduction of the fuel consumption. Simulations of the remaining systems and strategies did not show a reduced consumption. These results depend on our fuel consumption model, which could be improved by consideration of the gear as we mentioned in Section 8.4.1. However, many studies on congestion costs (see Chapter 1) point out highly increased fuel consumption because of congestion, thus prevention of congestion on its own reduces fuel consumption and pollution.

Objective 5 *It should work with limited equipment rates:*
Our simulations revealed that Jam-ADS in the closed system and continuous

Jam-ADS in the open system are effective with a limited equipment rate, which we tested down to an equipment rate of 5 %.

Objective 6 *It should be self-organizing:*
Jam-ADS is self-organizing by design. As long as a sufficient fraction of the drivers follow the recommendations of Jam-ADS, it reduces congestion without the need of any external services.

Objective 7 *It should increase the traffic capacity (maximum flow) of the road:*
In the closed system the average flow is proportional to the average velocity because of the fixed number of moving vehicles. Our simulations showed for the single-lane system a considerable increase of the average velocity by Jam-ADS and for the multiple-lane system a slight increase. Thus, in the closed system we observed increased flow.

In the open system we saw that Jam-ADS is able to achieve an outflow similar to the system's inflow, while without Jam-ADS the outflow is dramatically less than the inflow.

In all cases we observed free flow with Jam-ADS at densities at which massive congestion occurred without Jam-ADS. Thus, the maximum possible flow is increased.

Objective 8 *It should increase traffic safety or at least keep the current safety level:*
The current safety level is kept by design, as Jam-ADS never recommends to move faster than safely possible. On the contrary, if Jam-ADS is active it recommends slower driving, which increases safety.

Furthermore, Park and Saccomanno [113] found that a variation of average velocity between successive highway sections has a bad safety impact. Jam-ADS helps to reduce the variation of speed, thus it increases road safety.

Improving traffic safety is, of course, by itself an important goal of traffic engineering. However, there is an additional benefit as increased safety statistically reduces the number of accidents. As each accident causes a temporary bottleneck, increased safety increases the average traffic flow as well.

From these results we conclude that Jam-ADS improves the traffic flow on highways.

11.2. Outlook

Every scientific question answered by research raises more questions and suggests further research activities. The following lists some further approaches we would

like to follow next.

We did many simulations, which can always be refined and extended. We already mentioned improvements of the fuel consumption model. These could be extended to simulate noise emissions as well, because in urban areas vehicle noise is often considered along exhaust emissions. Another extension would be to simulate the wireless communication more realistically by applying the transmission models provided by Shawn or other network simulators.[1] Beside these extensions of the simulations, it would be interesting to simulate a real highway section and adapt the simulation parameters such that simulation runs without Jam-ADS resemble empirically-observed congestion and then test what happens if Jam-ADS is turned on.

Beside simulations of highway congestion, we would like to further develop the ideas on urban traffic in Section 10.2, implement them in a simulator, and test them thoroughly. Next, they should be integrated with Jam-ADS and applied to an urban road network.

The analytical investigations in Chapter 9 could be continued by applying Jam-ADS to a suitable macroscopic model. Averaging over the vehicles ahead could be accomplished by integration of velocity and density ahead.

The suggestions so far concerned further investigation of the Jam-ADS strategies proposed in this thesis. In addition, the strategies themselves might be further refined. First, Jam-ADS might detect and classify the traffic situation ahead and change its parameters depending on the situation. For example, instead of putting the HDC area into a fixed area upstream of a known bottleneck, like an on-ramp, the position and size of the HDC could depend on the current traffic situation. Recent insights into the development of congestion and the influence of bottlenecks on the shape of the fundamental diagram (see, for example, Coifman and Kim [20]) might help with this. Also, methods of fuzzy control could be beneficial to improve the dependency of Jam-ADS parameters on current data of the vehicles ahead.

Beyond these suggestions, the influence of Jam-ADS on a highway network should be investigated. Improved traffic flow in one part of a road network may cause congestion in another part that used to be uncongested. In a network, congestion or near-congestion information could be exchanged by C2C communication between the congestion hotspots and the strategy might adapt to that information.

We finish our outlook supporting an idea of Tom Vanderbilt, the author of a popular book about traffic [146]. He suggested in his blog[2] on March 9, 2010 to automatically make the mood of the music played in the car depending on the current traffic situation to calm down drivers stuck in a traffic jam. Beside all other benefits, Jam-ADS could be used to pass information about the current traffic situation to the music player to achieve this.

[1] We expect that a less reliable C2C communication has the same effect as a reduced equipment rate.

[2] `http://www.howwedrive.com/2010/03/09/the-mozart-effect-and-teen-driving`

A. CircSim as General Traffic Simulation Tool

CircSim was mainly developed for this work but with the intention to be flexible enough for other projects and future extensions. This appendix describes the general concepts and interfaces to adapt CircSim to other simulation tasks. See Section 6.3 for features referenced but not explained here. Also, many details are explained by comments in the code.

CircSim distinguishes the drive model and the lane-change model. Both are performed separately as two sub-steps of a simulation step. For both models CircSim distinguishes the traffic model itself and the influencers. The traffic model is assumed to simulate real vehicle drivers as the Krauß model does. The influencers are supposed to implement a specific behavioral strategy like Jam-ADS. For each vehicle in each sub-step, first the traffic model is applied to compute the new micro-state and then the influencers to its result. Several influencers could be applied in a chain. The separation of traffic models and influencers allows to replace traffic model and traffic-treating strategy independently to test the same strategy with different models and vice versa.

A.1. Data structures

The main data structure, `car`, contains all micro-states computed so far and grows unlimited during the simulation. For performance reasons the array is extended a number of steps ahead, because each extension causes costly operations inside MATLAB. `car` is organized as a struct of 2-dimensional arrays with time step numbers and vehicle numbers as indices, such that extending the number of time steps does not require reorganization of the order of the entries in memory.

All simulation configuration data is kept in a struct called `config`. The file `default_config.m` contains its initial values. This file is executed as MATLAB code, so assignments can contain any valid MATLAB expression. This even holds if CircSim was compiled with the MATLAB compiler. As the content of `default_config.m` is interpreted as regular MATLAB code, later assignments to the same variable override earlier assignments.

For easier organization of configurations the function

```
read_config(<config file name>)
```

allows to include further configuration files at the place of the command call. In fact the default configuration file `default_config.m` is also evaluated by calling `read_config()`. As explained, later assignments to the same variable override earlier assignments, so including a more special configuration file with `read_config(<config file name>)` at the end of `default_config.m` may override more general settings in `default_config.m`. CircSim comes with an example `default_config.m`, which contains some out-commented inclusions at the end applying this principle.

The struct `plot_marker` contains two elements `timecounts` and `labels` with a sequence of numbers and strings, respectively. These list special simulation steps where something extraordinary happened, like changing a configuration value. The assigned label is applied to mark these special simulation steps in plots with an appropriate legend entry.

All these data structures are kept in global variables to avoid massive copying for each change of the data. MATLAB lacks a reference data type, thus if a sub-function needs to change some passed data, it needs to return a copy of the data including the changes. The data structure `car` is changed for each simulation step and for each vehicle by several calls to sub-functions. Thus, copying it for each small change in a sub-function would cause a massive performance drop.

A.2. Main loop

There is no generic function to execute the main loop, because each simulation script (like the interactive dialog, for example) may do different things between the simulation steps. However, the function

```
simulation_step(timecount)
```

is provided to call model and influencer functions for each vehicle to compute the next simulation step. The main loop should call it with incrementing `timecount` to compute the micro-state after the passed `timecount`. Examples are in sub-function

```
button_run_Callback(...)
```

in file `simulation_gui.m` and the main function in `auto_simulation_run.m`.
`simulation_step(timecount)` executes the following tasks:

1. Extend arrays in `car` if there is no more free space left.
2. Compute new traffic lights state by calling `compute_traffic_lights_state()`.
3. Loop over all vehicles to compute their new velocities.

4. Loop over all vehicles to compute their preferred lanes.

5. Loop in random order over all vehicles that wish to change their lane, check if it is safe and perform the change if it turns out to be safe.

The main reason to split the computation of the preferred lane and the actual lane change is to make all lane changes depending on a consistent micro-state as explained in Section 6.3.1.

Another reason to separate the last two steps is to avoid an artificial vehicle number bias in lane changing. Otherwise, if two vehicles wish to change their lanes, but only one may safely do so, always the vehicle with the smaller number in the data structure `car` would be allowed to perform the lane change.

Before each of the three sub-steps computing the new micro-state, the function

```
init_sort_x_data(config, car, timecount, num_leaders)
```

is called to sort the vehicles both totally and per lane by their position and provide the resulting order information as returned struct. This information is passed to all traffic-model and influencer functions to avoid multiple sorting of the vehicles.

Looping over the vehicles to compute new velocities and preferred lanes has some similarities, hence sub-function

```
cars_loop(config, car, timecount, num_leaders, compute_car, model_data)
```

executes the loop generically. It calls either the sub-function

```
drive_car(config, cur_car, timecount, sort_x_data, model_data)
```

or

```
choose_lane(config, cur_car, timecount, sort_x_data, model_data)
```

for the specific code to compute the new state of a single vehicle. Both call the configured model functions first, and then all configured influencers, respectively. Additionally, the configured drive function of the traffic model could return special vehicle data, which are passed to the configured lane-change-model function. The Krauß-model implementation applies this to avoid double computation of v_{safe}.

As explained in Section 6.3.1 random deceleration needs to be executed after the influencers, which sub-function `drive_car()` does. Thus, the configured drive function of the traffic model should not perform random deceleration.

A.3. Plotting

Section 6.3.2 describes all types of plots CircSim can generate. Plotting is done up to a given simulation step in function

```
plot_simulation(max_timecount)
```

The data to plot is extracted from the global variable `car`. The current configuration stored in the global variable `config` determines which plots become rendered and how they look like.

The fuel consumptions for the fuel-consumption plot are computed as needed for the plot by calling

```
fuel_step(config, car, cur_car, timecount)
```

for each vehicle and simulation step. Depending on the settings in `config` it computes the absolute consumption of a vehicle during the given simulation step by part (acceleration, air resistance, rolling resistance, idle) as described in Section 6.3.1. Conversion to consumption per distance is done by the fuel-consumption plot function itself.

Additionally, `plot_simulation.m` contains some helper sub-functions to add general menus to several plots, like lane selection, for example. The helper function

```
draw_plot_markers(is_horizontal)
```

draws horizontal or vertical lines at the times contained in `plot_marker` described above and adds their labels to the legend.

A.4. Configuration settings

The `default_config.m` provided by CircSim contains all currently used configuration settings including short comments about their meanings and assigning reasonable values to them. However, we give some further explanations here on some settings that might be less obvious. See also Section 6.3 for more explanations. If not otherwise noted all values are set in SI units.[1] As CircSim is still under development there might be some obsolete configuration settings left.

`config.traffic_model.current` This should contain a struct with the elements `drive`, `choose_lane`, and `lane_change_safe` pointing to appropriate MATLAB functions. Sub-function `inner_config()` of `read_config.m` prepares those structs for the Krauß model and the IDM.

`config.acceleration_formula` Pointer to a MATLAB function to compute dynamically the current maximum acceleration of a vehicle depending on the simulation state.

[1] SI is the abbreviation for *Système international d'unités* or *International System of Units* and defines the basic physical units of the metric system.

`config.influencers.choose_lane` **and** `config.influencers.drive` Sequences of pointers to car-following and lane chage influencers.

`config.v_thresh` Threshold for congestion used by the Krauß model to decide about lane changing.

`config.car_classes` A struct with vehicle-type-specific settings. Each setting contains a list or sequence of equal length with the values for each vehicle type.

`config.traffic_lights` A struct with special settings for each traffic light. Like `config.car_classes` each struct element contains a list of equal length with the individual traffic light values. If all lists are empty there are no traffic lights.

`config.krauss_v_safe` CircSim can run the Krauß model with different expressions for v_{safe}. This setting points to a MATLAB function to compute v_{safe}.

`config.diagrams` List of initially checked plot types in the interactive dialog.

A.5. Simulation states

The files `save_simulation_state.m` and `load_simulation_state.m` contain functions to save and load the current simulation state contained in the global variables described in Appendix A.1. Additionally, a simulation state contains the current time count and the current state of the random number generator to ensure proper continuation of a simulation.

Beside that, the save function tries to determine the current commit reference if called from a Git repository and save it, as well. This allows to apply the correct CircSim revision to load an older simulation state if the current revision of CircSim is not able to load it any more. Additionally, it may help to debug inconsistent data by providing the exact code a simulation state was generated with.

B. TraciTypes

During the work for this thesis the TraCI interface of SUMO (Section 6.2.3) was replaced by a completely new protocol version. As a network protocol TraCI specifies exactly how to serialize the data of the TraCI commands into network messages. The commands contain data which needs to be encoded into network messages. TraCI defines many data types and their encoding into messages, which we call *TraCI-data-types*. Usually, but not always, an encoded TraCI-data-type is preceded by a one byte integer *TraCI-type-ID*.

The new TraCI protocol in SUMO required us to adapt Shawn's TraCI client implementation. To be prepared for future changes and extensions of the protocol we developed a flexible TraCI library based on C++ class templates implementing the new protocol.

TraCI-data-types are either *atomic* or *composed*. A composed TraCI-data-type consists of other TraCI-data-types, thus TraCI-data-types can be recursively combined to more complex types. In addition, some composed TraCI-data-types are dynamic-length lists of other TraCI-data-types. Such lists are encoded by an integer TraCI-data-type containing the number of elements followed by the elements itself. For more details about the TraCI protocol and the TraCI-data-types we refer to SUMO's TraCI documentation.[1]

In the following we distinguish TraCI-data-types and C++-types. The C++ standard [63, 64] defines *fundamental* and *compound* data types. Fundamental C++-types are provided by C++ itself (for example `int`) and compound types are defined by C++ code (`struct`, `class`, ...). Furthermore, the C++ standard defines that a region of storage having a type is called *object*. Thus, even a variable of type `int` is such a C++-object.

The source code of the TraciTypes library is part of Shawn's code[2] located in the subdirectory `src/apps/traci_client/`. The base classes and the main documentation is located in file `tracitype.h`. Most classes and their members are documented in the source code with *doxygen*[3].

[1] `http://sumo.sourceforge.net/doc/current/docs/userdoc/TraCI.html`

[2] Shawn's code is available via SVN under `https://shawn.svn.sourceforge.net/svnroot/shawn`. This appendix refers to revision 653.

[3] Doxygen is an open source tool to generate documentation out of specially formatted source code comments, available at `http://www.doxygen.org`.

B.1. Objectives

In our library each TraCI-data-type corresponds to a template instance derived from the common base class `TraciType`. Hence, we call the templates and their instances *TraciTypes*. They fulfill the following objectives:

- Within Shawn an instance of a TraciType should be usable like a C++-object of the type of the contained TraCI-data-type:
  - A TraciType corresponding to an atomic TraCI-data-type should be usable like a fundamental C++-type.
  - A TraciType corresponding to a composed TraCI-data-type should be usable like a struct in C++ with each contained TraciType as a public element variable.
  - A TraciType corresponding to a dynamic-length list TraCI-data-type should be usable as an STL container[4] with TraciType elements.

  See examples in the next section for explanation.
- Each TraciType should know, how to encode itself into a TraCI message and how to decode its data from a TraCI message, including optionally the TraCI-type-ID.
- TraciTypes should be easily combined to new TraciTypes corresponding to composed TraCI-data-types. Such composed TraciType should fulfill the above objectives as well.

We searched for an open programming library that conforms to these objectives at least without the TraCI-type-ID, but we did not find one. The closest we found were code generators based on ASN.1[5], but they were not sufficient for us.

B.2. Usage examples

To explain the objectives we give some examples from our library implementation. `IntegerType` is a TraciType corresponding to the atomic TraCI-data-type *integer*. It can be used like the C++-type `int`:

```
1 IntegerType x(message);  // read integer from TraCI message
2 int y = x + 1;           // use x transparently
3 x = 2 * y;               // assign new value to x
4 x.write(message2);       // write new x value into another message
```

[4] The *STL* is the standard template library of C++, which is part of the C++ standard. It defines several container types with a common interface to access their elements.

[5] *ASN.1* is an ISO-standardized notation for encoding of network data.

The next code example applies the TraciType `Position2DTypeWithTypeId`, which corresponds to the composed TraCI-data-type *2DPosition*. A *2DPosition* consists of two atomic TraCI-data-types *double* called x and y. Names of TraciTypes ending on "`WithTypeId`" read or write by default the corresponding TraCI-type-ID before the TraCI-data-type value from or to a TraCI message, thus in this case the value is read from `message` with a preceding TraCI-type-ID. In line 4 this behavior is turned off, thus the value is written to `message2` without a preceding TraCI-type-ID. `Position2DTypeWithTypeId` can be used like a `struct` with the elements `x` and `y` of type `double`:

```
1 Position2DTypeWithTypeId pos(message);  // read a 2DPosition
2                                         // including type ID
3 double x = pos.x;                       // get x coordinate
4 pos.setWithTypeId(false);               // turn off type ID
5 pos.write(message2);                    // write position without
6                                         // type ID into message2
```

We give a last example with dynamic-length TraciType `PolygonTypeWithTypeId`, which corresponds to the composed TraCI-data-type *Polygon*. A *Polygon* consists of an arbitrary number of *2DPosition* elements. The example computes the sum of the x-coordinates of the polygon points. `PolygonTypeWithTypeId` is derived from a `std::vector` with elements of type `Position2DType`, which are like `Position2DTypeWithTypeId`, except that they do not read or write the TraCI-type-ID:

```
1 PolygonTypeWithTypeId polygon(message); // read a polygon including
2                                         // type ID from message
3 double sumX = 0.0;                      // initialize sum
4 for(const auto &point : polygon)        // iterate over points
5     sumX += point.x;                    // update sum
```

This example makes use of the new range-based `for`-statement and of the new type-specifier `auto` of C++11 [64] to make the iteration code simpler.

B.3. Architecture

All TraciTypes are derived from the abstract base class `TraciType`, which is not a template, because templates cannot be used as polymorphic pointers. From `TraciType` we derive the template `GenTraciType` with a template parameter `MyMessageHandler`. `GenTraciType` expects a class derived from the abstract interface `MessageHandler` as `MyMessageHandler`. A `MessageHandler` contains the

C++-type corresponding to the TraCI-data-type and member functions for reading or writing the data from or into a TraCI message, respectively.

The `MessageHandler` is the backend of a TraciType, because it contains all specific properties of a certain TraCI-data-type while `GenTraciType` is the generic frontend with a comfortable common interface to all TraciTypes. This architecture makes it easy to implement new TraCI-data-types as TraciTypes, because only the simple interface of class `MessageHandler` has to be implemented.

The TraciTypes library provides a specialization of `MessageHandler` called `MessageHandlerComposed` to easily implement a composed TraCI-data-type, and a further specialization of `MessageHandlerComposed` called `MessageHandlerDynamicLength` to easily implement a dynamic-length-list TraCI-data-type.

To realize the objective of transparent usage of the TraCI data as C++-type the library contains two specializations of `GenTraciType`, one for fundamental C++-types and one for compound C++-types. `FundamentalTraciType` contains a type conversion constructor and a type conversion operator returning a reference to enable transparent access to the contained fundamental C++-type. For example, `IntegerType` in the first code example of Section B.2 is a template instance of `FundamentalTraciType`. The access to `x` in lines 2 and 3 calls implicitly the type conversion operator of `FundamentalTraciType`.

The specialization `CompoundTraciType` of `GenTraciType` is realized as additional public derivation from a C++ struct, which contains the elements of the corresponding composed TraCI-data-type as members. The derivation enables direct access of these members through the TraciType as in the second example of Section B.2 with `Position2DTypeWithTypeId`. The dynamic-length TraciType `PolygonTypeWithTypeId` in the third example of Section B.2 is also an instance of `CompoundTraciType`. It is derived from a `std::vector` containing the elements of TraCI's dynamic length list.

B.4. Examples of adding more TraciTypes

The following example demonstrates how simple it is to implement a composed TraCI-data-type as a TraciType class. The example reproduces the original library code of `Position2dTypeWithTypeId` in the second example of Section B.2, except that a single `typedef` in the original code is split into several `typedef`s here, for easier readability:

```
1 struct Position2d
2 {
3     DoubleType x;
4     DoubleType y;
5
```

```
6  protected:
7      // set order of elements in TraCI type
8      virtual void init(ComposedContainer &queue)
9      {
10         queue.push_back(&x);
11         queue.push_back(&y);
12     }
13 };
14 typedef ComposedValue<Position2d>  Position2dValue;
15 typedef MessageHandlerComposed<Position2dValue, id::Position2d>
16             Position2dMessageHandler;
17 typedef CompoundTraciType<Position2dMessageHandler, true>
18             Position2dTypeWithTypeId;
```

The struct `Position2d` defines the compound C++-type containing the data of the TraciType. It is composed of two elements of the atomic TraCI-data-type *double*, which are implemented as TraciType `DoubleType`. In addition, the struct contains a member function `init`, which sets the order in which the data elements are encoded in a TraCI message by pushing references to the data elements into a passed container named `queue`. We had to add the `init`-function, because C++ provides no direct possibility to access the order of class or struct elements at run-time.[6]

The template `ComposedValue` instantiated in line 14 adds code to `Position2d` to manage the order of elements defined in member function `init`, besides several constructors, comparison operators, and an assignment operator. Those are needed by the template `MessageHandlerComposed` to instantiate a particular `MessageHandler` in line 15/16. The second template argument, `id::Position2d`, of `MessageHandlerComposed` is the TraCI-type-ID of the corresponding TraCI-data-type *2DPosition*. Finally in line 17/18 the new `MessageHandler` is applied to instantiate a new `CompoundTraciType`. The boolean template argument `true` tells the template `CompoundTraciType` that the data of `Position2dTypeWithTypeId` is by default encoded with a preceding TraCI-type-ID as denoted by the ending "`WithTypeId`".

The next example demonstrates the implementation of a dynamic-length-list TraCI-data-type.

```
1 typedef CompoundTraciType<Position2dMessageHandler, false>
2             Position2dType;
```

[6] Data elements of classes are initialized in the order they are declared in the class definition and it might be possible to record this order of initialization at run-time. Maybe this could be applied to implicitly perform the task of the `init`-function, which would be a further improvement of the library.

```
3 typedef DynamicLengthValue<Position2dType, UbyteType>
4             PolygonPointContainer;
5 typedef MessageHandlerDynamicLength<PolygonPointContainer,
6             id::TypePolygon>  MessageHandlerPolygon;
7 typedef CompoundTraciType<MessageHandlerPolygon, true>
8             PolygonTypeWithTypeId;
```

Line 1/2 is the same as line 17/18 of the previous example except that the resulting TraciType, `Position2dType`, encodes its data without a preceding TraCI-type-ID. The argument `false` is omitted in the original code, because it is set as default value of the template parameter. Line 3/4 uses `Position2dType` as element type of a dynamic-length list whose length is encoded as TraCI-data-type *ubyte*. `PolygonPointContainer` is the value type for a particular dynamic length list, `MessageHandler`, instantiated in line 5/6 with TraCI-type-ID `id::TypePolygon`. Finally in line 7/8, the new `MessageHandler` is applied to instantiate the TraciType for polygons.

The presented examples have shown that the TraciTypes library, which we added to Shawn, fulfills the required objectives listed in Section B.1, mainly the transparent access of the contained values and the easy extensibility of the existing TraciTypes.

C. Proof of Symmetry of Bode Plots

In Section 9.4.3 we stated that the absolute of the transfer function of a linear system after $\mathcal{Z}$-transform is symmetric around integer multiples of $\pi/\Delta t$ and its phase angle is anti-symmetric. We prove this here.

In the following, we denote a complex conjugate by a line above, like $\overline{z}$. First, we prove a simple theorem about polynomials with real coefficients.

Theorem. *Let $p : \mathbb{C} \to \mathbb{C}$ be a polynomial with real coefficients and $z \in \mathbb{C}$. Then*

$$\overline{p(z)} = p(\overline{z}) \quad . \tag{C.1}$$

Proof. According to the complex conjugate root theorem the zeros of p are either real or pairs of complex conjugates. Suppose r_i with $i = 1, \ldots, n$ are the real roots and c_j with $j = 1, \ldots, m$ are one element of each pair of complex roots. Then the fundamental theorem of algebra allows us to write

$$p(\overline{z}) = \prod_{i=1}^{n} (r_i - \overline{z}) \prod_{j=1}^{m} (c_j - \overline{z})(\overline{c_j} - \overline{z}) \tag{C.2}$$

$$= \prod_{i=1}^{n} \overline{(r_i - z)} \prod_{j=1}^{m} \overline{(\overline{c_j} - z)(c_j - z)} \tag{C.3}$$

$$= \overline{p(z)} \quad . \tag{C.4}$$

□

The transfer function of a discrete linear system always has the form of a rational function

$$G(z) = \frac{b_m z^m + \cdots + b_1 z + b_0}{z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0} \tag{C.5}$$

with real coefficients. The coefficients are real, because they are the same as in the original linear system equations in time domain.

Applying the theorem above we get

$$G(\overline{z}) = \overline{G(z)} \quad . \tag{C.6}$$

Inserting the substitution $z = e^{i\omega\Delta t}$ yields

$$G(z(-\omega)) = G(\overline{z(\omega)}) = \overline{G(z(\omega))} \quad . \tag{C.7}$$

Thus, the absolute of transfer function G is symmetric and the phase angle anti-symmetric:

$$|G(z(-\omega))| = |G(z(\omega))| \tag{C.8}$$

$$\arg(G(z(-\omega))) = -\arg(G(z(\omega))) \quad . \tag{C.9}$$

Together with the $2\pi/\Delta t$ periodicity of z we have the same symmetries around all integer multiples of $\pi/\Delta t$.

List of Figures

List of Tables

Bibliography

[1] M. Appl and R. Sollacher. *Intelligente Verkehrsbeeinflussung mit adaptiven, lernenden Systemen (Intelligent Traffic Control with Adaptive, Learning Systems)*. In: *Automatisierungstechnik* 49.11 (Nov. 2001), p. 512. DOI: 10.1524/auto.2001.49.11.512 (cit. on p. 205).

[2] Roberto Baldessari, Bert Bödekker, Matthias Deegener, Andreas Festag, Walter Franz, C. Christopher Kellum, Timo Kosch, Andras Kovacs, Massimiliano Lenardi, Cornelius Menig, Timo Peichl, Matthias Röckl, Dieter Seeberger, Markus Straßberger, Hannes Stratil, Hans-Jörg Vögel, Benjamin Weyl, and Wenhui Zhang. *CAR 2 CAR Communication Consortium Manifesto (Version 1.1)*. Ed. by Car-2-Car Communication Consortium. Dublin, Ireland, Aug. 2007 (cit. on p. 32).

[3] M. Bando, K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama. *Structure stability of congestion in traffic dynamics*. In: *Japan Journal of Industrial and Applied Mathematics* 11 (1994), pp. 203–223. ISSN: 0916-7005. DOI: 10.1007/BF03167222 (cit. on p. 7).

[4] M. Bando, K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama. *Dynamical model of traffic congestion and numerical simulation*. In: *Phys. Rev. E* 51 (Feb. 1995), pp. 1035–1042. DOI: 10.1103/PhysRevE.51.1035 (cit. on pp. 19, 20).

[5] Milan Batista and Elen Twrdy. *Optimal velocity functions for car-following models*. In: *Journal of Zhejiang University – Science A* 11 (7 2010), pp. 520–529. ISSN: 1673-565X. DOI: 10.1631/jzus.A0900370 (cit. on p. 20).

[6] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. *SUMO – Simulation of Urban MObility – an overview*. In: *SIMUL 2011, the third international conference on advances in system simulation*. Barcelona, Spain, Oct. 2011, pp. 55–60. ISBN: 978-1-61208-169-4 (cit. on p. 44).

[7] Dietrich Braess. *Über ein Paradoxon aus der Verkehrsplanung*. In: *Mathematical Methods of Operations Research* 12.1 (Dec. 1968), pp. 258–268 (cit. on p. 30).

[8] Dietrich Braess. *On a paradox of traffic planning*. In: *Transportation Science* 39.4 (Nov. 2005), pp. 446–450 (cit. on p. 30).

[9] Werner Brilon. *Traffic engineering and the new German highway capacity manual*. In: *Transportation Research Part A: Policy and Practice* 28.6 (1994), pp. 469–481. ISSN: 0965-8564. DOI: 10.1016/0965-8564(94)90045-0 (cit. on p. 125).

[10] Werner Brilon, Justin Geistefeldt, and Matthias Regler. *Reliability of freeway traffic flow: a stochastic concept of capacity*. In: *Proceedings of the 16th international symposium on transportation and traffic theory*. College Park, Maryland, July 2005, pp. 125–144 (cit. on p. 7).

[11] Elmar Brockfeld, Reinhart Kühne, and Peter Wagner. *Calibration and validation of microscopic traffic flow models*. In: *Transportation Research Board 2005*. Vol. 1934. Transportation Research Board of the National Academies, Feb. 2005, pp. 179–187. DOI: 10.3141/1934-19 (cit. on p. 21).

[12] Elmar Brockfeld and Peter Wagner. *Testing and benchmarking of microscopic traffic flow models*. In: *WCTR04 – 10th World Conference on Transport Research*. Istanbul (Turkey), July 2004, pp. 775–776 (cit. on p. 21).

[13] Bureau of Transport and Regional Economics. *Estimating urban traffic and congestion cost trends for Australian cities*. Tech. rep. Working Paper 71. Canberra, Australia: Department of Transport and Regional Services, Apr. 2007 (cit. on p. 2).

[14] R.C. Carlson, I. Papamichail, and M. Papageorgiou. *Comparison of local feedback controllers for the mainstream traffic flow on freeways using variable speed limits*. In: *14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. Oct. 2011, pp. 2160–2167. DOI: 10.1109/ITSC.2011.6082967 (cit. on p. 28).

[15] R.C. Carlson, I. Papamichail, and M. Papageorgiou. *Local feedback-based mainstream traffic flow control on motorways using variable speed limits*. In: *Intelligent Transportation Systems, IEEE Transactions on* 12.4 (Dec. 2011), pp. 1261–1276. ISSN: 1524-9050. DOI: 10.1109/TITS.2011.2156792 (cit. on p. 28).

[16] M. J. Cassidy and Michael Mauch. *An observed traffic pattern in long freeway queues*. In: *Transportation Research Part A: Policy and Practice* 35.2 (2001), pp. 143–156. ISSN: 0965-8564. DOI: 10.1016/S0965-8564(99)00052-X (cit. on p. 9).

[17] Debashish Chowdhury, Ludger Santen, and Andreas Schadschneider. *Statistical physics of vehicular traffic and some related systems*. In: *Physics Reports* 329.4–6 (May 2000), pp. 199–329 (cit. on pp. 5, 11).

[18] R. Chrobok, A. Pottmeier, S. F. Marinósson, and M. Schreckenberg. *Online simulation and traffic forecast: applications and results.* In: *Proceedings of the Internet and Multimedia Systems and Applications.* 2002, pp. 113–118 (cit. on p. 29).

[19] Li Chuan-Yao, Tang Tie-Qiao, Huang Hai-Jun, and Shang Hua-Yan. *A new car-following model with consideration of driving resistance.* In: *Chinese Physics Letters* 28.3 (2011), p. 038902 (cit. on p. 20).

[20] Benjamin Coifman and Seoungbum Kim. *Extended bottlenecks, the fundamental relationship, and capacity drop on freeways.* In: *Transportation Research Part A: Policy and Practice* 45.9 (2011), pp. 980–991. ISSN: 0965-8564. DOI: 10.1016/j.tra.2011.04.003 (cit. on pp. 7, 213).

[21] David N. Cottingham and Jonathan J. Davies. *A vision for wireless access on the road network.* In: *Proceedings of the 4th International Workshop on Intelligent Transportation (WIT 2007).* Hamburg, Germany, Mar. 2007, pp. 25–30 (cit. on p. 33).

[22] David N. Cottingham, Jonathan J. Davies, and Alastair R. Beresford. *Congestion-aware vehicular traffic routing using WiFi hotspots.* In: *Proceedings of the Communications Innovation Institute Workshop.* Cambridge, UK, May 2005 (cit. on p. 33).

[23] L.C. Davis. *Mitigation of congestion at a traffic bottleneck with diversion and lane restrictions.* In: *Physica A: Statistical Mechanics and its Applications* 391.4 (2012), pp. 1679–1691. ISSN: 0378-4371. DOI: 10.1016/j.physa.2011.10.036 (cit. on pp. 20, 29).

[24] Li-yun Dong, Xu-dan Weng, and Qing-ding Li. *Velocity anticipation in the optimal velocity model.* In: *Journal of Shanghai University (English Edition)* 13 (4 2009), pp. 327–332. ISSN: 1007-6417. DOI: 10.1007/s11741-009-0415-3 (cit. on p. 20).

[25] Sebastian Ebers, Sándor P. Fekete, Stefan Fischer, Horst Hellbrück, Björn Hendriks, and Axel Wegener. *Hovering data clouds for organic computing.* In: *Organic Computing — A Paradigm Shift for Complex Systems.* Ed. by Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer. Vol. 1. Autonomic Systems. Springer Basel, 2011, pp. 221–234. ISBN: 978-3-0348-0129-4. DOI: 10.1007/978-3-0348-0130-0_14 (cit. on pp. 35–38).

[26] Sebastian Ebers, Mohamed A. Hail, Stefan Fischer, and Horst Hellbrück. *API for data dissemination protocols – evaluation with AutoCast.* In: *The Third World Congress on Nature and Biologically Inspired Computing (NaBIC 2011).* Salamanca, Spain: IEEE, Oct. 2011, pp. 534–539. DOI: 10.1109/NaBIC.2011.6089644 (cit. on p. 38).

[27] Stephan Eichler, Benedikt Ostermaier, Christoph Schroth, and Timo Kosch. *Simulation of car-to-car messaging: analyzing the impact on road traffic.* In: *Proceedings of the 13th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE Computer Society, Sept. 2005, pp. 507–510 (cit. on p. 33).

[28] M. R. Evans, N. Rajewsky, and E. R. Speer. *Exact solution of a cellular automaton for traffic.* In: *Journal of Statistical Physics* 95 (1 1999), pp. 45–96. ISSN: 0022-4715. DOI: 10.1023/A:1004521326456 (cit. on p. 14).

[29] Sándor P. Fekete, Björn Hendriks, Christopher Tessars, Axel Wegener, Horst Hellbrück, Stefan Fischer, and Sebastian Ebers. *Methods for improving the flow of traffic.* In: *Organic Computing — A Paradigm Shift for Complex Systems*. Ed. by Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer. Vol. 1. Autonomic Systems. Springer Basel, 2011, pp. 447–460. ISBN: 978-3-0348-0129-4. DOI: 10.1007/978-3-0348-0130-0_29 (cit. on p. 39).

[30] Sándor P. Fekete, Alexander Kröller, Stefan Fischer, and Dennis Pfisterer. *Shawn: the fast, highly customizable sensor network simulator.* In: *Proceedings of the 4th International Conference on Networked Sensing Systems (INSS)*. Vol. 6-8. June 2007, p. 299. DOI: 10.1109/INSS.2007.4297441 (cit. on p. 48).

[31] Sándor P. Fekete, Christiane Schmidt, Axel Wegener, and Stefan Fischer. *Hovering data clouds for recognizing traffic jams.* In: *Proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISOLA)*. 2006, pp. 213–218 (cit. on p. 36).

[32] Sándor P. Fekete, Christiane Schmidt, Axel Wegener, Horst Hellbrück, and Stefan Fischer. *Empowered by wireless communication: distributed methods for self-organizing traffic collectives.* In: *ACM Transact. Autonom. and Adaptive Syst.* 5.3 (2010), pp. 439–462. DOI: 10.1145/1837909.1837912 (cit. on p. 35).

[33] Andreas Festag, Holger Füßler, Hannes Hartenstein, Amardeo Sarma, and Ralf Schmitz. *FleetNet: bringing car-to-car communication into the real world.* In: *Proceedings of the 11th World Congress on ITS*. Nagoya, Japan, Oct. 2004 (cit. on p. 32).

[34] Andreas Festag, Gerhard Noecker, Markus Strassberger, Andreas Lübke, Bernd Bochow, Marc Torrent-Moreno, Sascha Schnaufer, Robert Eigner, Catrinel Catrinescu, and Jürgen Kunisch. *'NoW – network on wheels': project objectives, technology and achievements.* In: *Proceedings of the 5th International Workshop on Intelligent Transportation (WIT)*. Hamburg, Germany, Mar. 2008, pp. 211–216 (cit. on p. 32).

[35] Lisa Fleischer and Martin Skutella. *Quickest flows over time*. In: *SIAM Journal on Computing* 36.6 (2007), pp. 1600–1630. DOI: 10.1137/S0097539703427215 (cit. on p. 40).

[36] Gunnar Flötteröd. *Cadyts – a free calibration tool for dynamic traffic simulations*. In: *9th STRC Swiss Transport Research Conference*. Sept. 2009 (cit. on p. 44).

[37] Walter Franz and Hannes Hartenstein. *Inter-Vehicle-Communications Based on Ad Hoc Networking Principles – The FleetNet Project*. Ed. by Martin Mauve. Universitätsverlag Karlsruhe, Karlsruhe, 2005. ISBN: 3-937300-88-0 (cit. on p. 32).

[38] Jochen Fromm and Michael Zapf. *„Selbst"-Eigenschaften in verteilten Systemen*. In: *Praxis der Informationsverarbeitung und Kommunikation (PIK)* 28.4 (Dec. 2005), pp. 198–198. ISSN: 0930-5157. DOI: 10.1515/PIKO.2005.189 (cit. on p. 40).

[39] Justin Geistefeldt. *Capacity effects of variable speed limits on German freeways*. In: *Procedia – Social and Behavioral Sciences* 16 (2011), pp. 48–56. ISSN: 1877-0428. DOI: 10.1016/j.sbspro.2011.04.428 (cit. on p. 27).

[40] P. G. Gipps. *A behavioural car-following model for computer simulation*. In: *Transportation Research Part B: Methodological* 15.2 (1981), pp. 105–111. ISSN: 0191-2615. DOI: 10.1016/0191-2615(81)90037-0 (cit. on p. 14).

[41] P. G. Gipps. *A model for the structure of lane-changing decisions*. In: *Transportation Research Part B: Methodological* 20.5 (1986), pp. 403–414. ISSN: 0191-2615. DOI: 10.1016/0191-2615(86)90012-3 (cit. on p. 23).

[42] Phil Goodwin. *The Economic Costs of Road Traffic Congestion*. Tech. rep. ESRC Transport Studies Unit, University College London, May 2004 (cit. on p. 2).

[43] *GNU General Public License*. Free Software Foundation. June 29, 2007. URL: http://www.gnu.org/licenses/gpl.html (cit. on pp. 44, 78).

[44] Victor Gradinescu, Cristian Gorgorin, Raluca Diaconescu, Valentin Cristea, and Liviu Iftode. *Adaptive traffic lights using car-to-car communication*. In: *Proceedings of the 65th IEEE Vehicular Technology Conference Spring 2007 (VTC 2007-Spring)*. Dublin, Ireland, Apr. 2007, pp. 21–25. DOI: 10.1109/VETECS.2007.17 (cit. on p. 205).

[45] Greater Toronto Transportation Authority. *Costs of Road Congestion in the Greater Toronto and Hamilton Area: Impact and Cost Benefit Analysis of the Metrolinx Draft Regional Transportation Plan*. Tech. rep. Dec. 2008 (cit. on p. 2).

[46] G. Guo and W. Yue. *Hierarchical platoon control with heterogeneous information feedback*. In: *Control Theory & Applications, IET* 5.15 (Oct. 2011), pp. 1766–1781. ISSN: 1751-8644. DOI: 10.1049/iet-cta.2010.0765 (cit. on p. 31).

[47] H. Hao and P. Barooah. *On achieving size-independent stability margin of vehicular lattice formations with distributed control*. In: *arXiv.org* (Aug. 2011). arXiv:1108.1844 (cit. on p. 31).

[48] He Hao and Prabir Barooah. *Stability and robustness of large vehicular platoons with linear and nonlinear decentralized control for two architectures*. Submitted to Internatiol Journal of Robust and Nonlinear Control. 2012 (cit. on p. 31).

[49] A. Hegyi, T. Bellemans, and B. De Schutter. *Freeway traffic management and control*. In: *Encyclopedia of Complexity and Systems Science*. Ed. by R. A. Meyers. New York, NY: Springer, 2009, pp. 3943–3964. DOI: 10.1007/978-0-387-30440-3_232 (cit. on pp. 5, 25, 27, 29–31).

[50] A. Hegyi, B. De Schutter, and H. Hellendoorn. *Model predictive control for optimal coordination of ramp metering and variable speed limits*. In: *Transportation Research Part C* 13.3 (June 2005), pp. 185–209. DOI: 10.1016/j.trc.2004.08.001 (cit. on pp. 27, 28).

[51] Dirk Helbing. *Traffic and related self-driven many-particle systems*. In: *Rev. Mod. Phys.* 73.4 (Dec. 2001), pp. 1067–1141. DOI: 10.1103/RevModPhys.73.1067 (cit. on pp. 5, 40, 123, 151).

[52] Dirk Helbing, Ansgar Hennecke, Vladimir Shvetsov, and Martin Treiber. *Micro- and macrosimulation of freeway traffic*. In: *Mathematical and Computer Modelling* 35 (2002), p. 517. arXiv:cond-mat/0003269 (cit. on p. 11).

[53] Dirk Helbing and Benno Tilch. *Generalized force model of traffic dynamics*. In: *Phys. Rev. E* 58 (July 1998), pp. 133–138. DOI: 10.1103/PhysRevE.58.133 (cit. on p. 20).

[54] D. Helbing and M. Moussaid. *Analytical calculation of critical perturbation amplitudes and critical densities by non-linear stability analysis of a simple traffic flow model*. In: *The European Physical Journal B - Condensed Matter and Complex Systems* 69 (4 2009), pp. 571–581. ISSN: 1434-6028. DOI: 10.1140/epjb/e2009-00042-6 (cit. on p. 20).

[55] Horst Hellbrück, Axel Wegener, Junjian Cao, and Tian Zuo. *Fast prototyping for VANET applications with PDAs*. In: *Proceedings of the 1st international conference on wireless communication society, vehicular technology, information theory and aerospace & electronic systems technology (wireless VITAE)*. Aalborg, Denmark, May 2009 (cit. on p. 35).

[56] Horst Hellbrück, Axel Wegener, and Stefan Fischer. *AutoCast: a general-purpose data dissemination protocol and its application in vehicular networks*. In: *Ad Hoc & Sensor Wireless Networks Journal (AHSWN)* 6.1–2 (2008), pp. 91–122. ISSN: 1551-9899 (cit. on p. 38).

[57] Klaus Herrmann, Matthias Werner, and Gero Mühl. *A methodology for classifying self-organizing software systems*. In: *International Transactions on Systems Science and Applications* 2.1 (2006), pp. 41–50 (cit. on p. 40).

[58] Tsin Hing Heung, Tin Kin Ho, and Yu Fai Fung. *Coordinated road-junction traffic control by dynamic programming*. In: *Intelligent Transportation Systems, IEEE Transactions on* 6.3 (Sept. 2005), pp. 341–350. ISSN: 1524-9050. DOI: 10.1109/TITS.2005.853713 (cit. on p. 205).

[59] Serge P. Hoogendoorn and Piet H. L. Bovy. *State-of-the-art of vehicular traffic flow modelling*. In: *The Delft University of Technology, Delft*. Vol. 215. 4. 2001, pp. 283–303. DOI: 10.1177/095965180121500402 (cit. on p. 11).

[60] R. Hoyer and M. Fellendorf. *Parametrization of microscopic traffic flow models through image processing*. In: *Transportation Systems 1997*. Ed. by M. Papageorgiou and A. Pouliezos. Vol. 2. 1997, pp. 889–894 (cit. on p. 13).

[61] Dijiang Huang, Swaroop Shere, and Soyoung Ahn. *Dynamic highway congestion detection and prediction based on shock waves*. In: *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking*. VANET '10. New York, NY, USA: ACM, Sept. 2010, pp. 11–20. ISBN: 978-1-4503-0145-9. DOI: 10.1145/1860058.1860061 (cit. on p. 26).

[62] Haijun Huang, Tieqiao Tang, and Ziyou Gao. *Continuum modeling for two-lane traffic flow*. In: *Acta Mechanica Sinica* 22 (2006), pp. 131–137. ISSN: 0567-7718. DOI: 10.1007/s10409-006-0101-y (cit. on p. 23).

[63] *ISO/IEC 14882:2003: Programming languages — C++*. American National Standards Institute, Dec. 2003 (cit. on pp. 44, 221).

[64] *ISO/IEC 14882:2011: Programming languages — C++*. American National Standards Institute, Feb. 2012 (cit. on pp. 44, 118, 221, 223).

[65] Kitae Jang and Michael J. Cassidy. *Dual influences on vehicle speed in special-use lanes and critique of US regulation*. In: *Transportation Research Part A: Policy and Practice* 46.7 (2012), pp. 1108–1123. ISSN: 0965-8564. DOI: 10.1016/j.tra.2012.01.008 (cit. on p. 10).

[66] Stefan Joerer, Christoph Sommer, and Falko Dressler. *Toward reproducibility and comparability of IVC simulation studies: a literature survey*. In: *IEEE Communications Magazine* 50.10 (Oct. 2012), pp. 82–88. DOI: 10.1109/MCOM.2012.6316780 (cit. on p. 44).

[67] Dominic Jost and Kai Nagel. *Probabilistic traffic flow breakdown in stochastic car following models*. In: *arXiv.org* (2002). arXiv:cond-mat/0208082 (cit. on pp. 12, 16).

[68] Boris S. Kerner. *Experimental features of self-organization in traffic flow*. In: *Phys. Rev. Lett.* 81 (Oct. 1998), pp. 3797–3800. DOI: 10.1103/PhysRevLett.81.3797 (cit. on pp. 7, 40, 74).

[69] Boris S. Kerner. *Control of spatiotemporal congested traffic patterns at highway bottlenecks*. In: *Physica A: Statistical Mechanics and its Applications* 355.2–4 (2005), pp. 565–601. ISSN: 0378-4371. DOI: 10.1016/j.physa.2005.04.025 (cit. on p. 27).

[70] Boris S. Kerner. *A theory of traffic congestion at heavy bottlenecks*. In: *Journal of Physics A: Mathematical and Theoretical* 41.21 (2008), 215101 (42pp) (cit. on p. 9).

[71] Boris S. Kerner. *Effect of driver behavior on spatiotemporal congested traffic patterns at highway bottlenecks in the framework of three-phase traffic theory*. In: *arXiv.org* (Dec. 2010). arXiv:1012.5159 (cit. on p. 31).

[72] Boris S. Kerner and Sergey L. Klenov. *A microscopic model for phase transitions in traffic flow*. In: *Journal of Physics A: Mathematical and General* 35.3 (2002), p. L31 (cit. on p. 20).

[73] Boris S. Kerner and Sergey L. Klenov. *Microscopic theory of spatial-temporal congested traffic patterns at highway bottlenecks*. In: *Phys. Rev. E* 68.3 (Sept. 2003), p. 036130. DOI: 10.1103/PhysRevE.68.036130 (cit. on p. 9).

[74] Boris S. Kerner and Sergey L. Klenov. *Deterministic approach to microscopic three-phase traffic theory*. In: *MATH.GEN.* 39 (2006), p. 1775 (cit. on p. 21).

[75] Boris S. Kerner and Sergey L. Klenov. *Phase transitions in traffic flow on multilane roads*. In: *Phys. Rev. E* 80.5 (Nov. 2009), p. 056101. DOI: 10.1103/PhysRevE.80.056101 (cit. on p. 9).

[76] Boris S. Kerner, Sergey L. Klenov, and A. Brakemeier. *Testbed for wireless vehicle communication: a simulation approach based on three-phase traffic theory*. In: *arXiv.org* (Dec. 2007). arXiv:0712.2711 (cit. on p. 33).

[77] Boris S. Kerner, Sergey L. Klenov, Andreas Hiller, and Hubert Rehborn. *Microscopic features of moving traffic jams*. In: *arXiv.org* (Oct. 2005). arXiv:physics/0510167 (cit. on p. 9).

[78] B. S. Kerner and H. Rehborn. *Experimental features and characteristics of traffic jams*. In: *Phys. Rev. E* 53 (Feb. 1996), R1297–R1300. DOI: 10.1103/PhysRevE.53.R1297 (cit. on pp. 8, 74).

[79] B. S. Kerner and H. Rehborn. *Experimental properties of complexity in traffic flow*. In: *Phys. Rev. E* 53 (May 1996), R4275–R4278. DOI: 10.1103/PhysRevE.53.R4275 (cit. on pp. 7, 9).

[80] Arne Kesting, Martin Treiber, and Dirk Helbing. *Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity*. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 368.1928 (2010), pp. 4585–4605. DOI: 10.1098/rsta.2010.0084 (cit. on p. 31).

[81] Arne Kesting, Martin Treiber, Martin Schönhof, and Dirk Helbing. *Adaptive cruise control design for active congestion avoidance*. In: *Transportation Research Part C: Emerging Technologies* 16.6 (2008), pp. 668–683. ISSN: 0968-090X. DOI: 10.1016/j.trc.2007.12.004 (cit. on p. 31).

[82] A. Kesting, M. Treiber, and D. Helbing. *Agents for traffic simulation*. In: *arXiv.org* (May 2008). arXiv:0805.0300 (cit. on pp. 19, 33).

[83] Youngho Kim and Hartmut Keller. *Zur Dynamik zwischen Verkehrszuständen im Fundamentaldiagramm*. In: *Straßenverkehrstechnik* 45.9 (2001), pp. 433–442. ISSN: 0039-2219 (cit. on pp. 5, 8, 9).

[84] F. Knorr, D. Baselt, M. Schreckenberg, and M. Mauve. *Reducing traffic jams via VANETs*. In: *Vehicular Technology, IEEE Transactions on* PP.99 (2012), p. 1. ISSN: 0018-9545. DOI: 10.1109/TVT.2012.2209690 (cit. on pp. 14, 34).

[85] Florian Knorr and Michael Schreckenberg. *Influence of inter-vehicle communication on peak hour traffic flow*. In: *Physica A: Statistical Mechanics and its Applications* (2011), ISSN: 0378-4371. DOI: 10.1016/j.physa.2011.11.027 (cit. on p. 34).

[86] Ekkehard Köhler and Martin Skutella. *Flows over time with load-dependent transit times*. In: *SIAM Journal on Optimization* 15.4 (2005), pp. 1185–1202. DOI: 10.1137/S1052623403432645 (cit. on p. 40).

[87] Daniel Krajzewicz, Michael Bonert, and Peter Wagner. *The open source traffic simulation package SUMO*. In: *RoboCup 2006 infrastructure simulation competition*. Bremen, Germany, 2006 (cit. on p. 44).

[88] Stefan Krauß. *Microscopic modeling of traffic flow: investigation of collision free vehicle dynamics*. PhD thesis. Hauptabteilung Mobilität und Systemtechnik des DLR Köln, Apr. 1998 (cit. on pp. 15, 22).

[89] Stefan Krauß, Peter Wagner, and Christian Gawron. *Metastable states in a microscopic model of traffic flow*. In: *Physical Review E* 55.304 (May 1997), p. 5597 (cit. on pp. 21, 52, 57).

[90] A. Kröller, D. Pfisterer, C. Buschmann, S. P. Fekete, and S. Fischer. *Shawn: a new approach to simulating wireless sensor networks*. In: *Design, analysis, and simulation of distributed systems 2005, part of the SpringSim 2005*. Apr. 2005, pp. 117–124 (cit. on p. 48).

[91] Reinhart D. Kühne. *Foundations of traffic flow theory I: Greenshields' legacy – highway traffic*. In: *Symposium on the Fundamental Diagram: 75 Years*. 2008, p. 3 (cit. on p. 5).

[92] Hyun Keun Lee, Robert Barlovic, Michael Schreckenberg, and Doochul Kim. *Mechanical restriction versus human overreaction triggering congested traffic states*. In: *Phys. Rev. Lett.* 92 (June 2004), p. 238702. DOI: 10.1103/PhysRevLett.92.238702 (cit. on p. 34).

[93] Hyun Keun Lee and Beom Jun Kim. *Dissolution of traffic jam via additional local interactions*. In: *arXiv.org* (Nov. 2011). arXiv:1109.2191 (cit. on p. 34).

[94] Hyun Keun Lee and Beom Jun Kim. *Dissolution of traffic jam via additional local interactions*. In: *Physica A: Statistical Mechanics and its Applications* 390.23–24 (Nov. 2011), pp. 4555–4561. ISSN: 0378-4371. DOI: 10.1016/j.physa.2011.07.033 (cit. on p. 33).

[95] Suzanne E. Lee, Erik C. B. Olsen, and Walter W. Wierwille. *A Comprehensive Examination of Naturalistic Lane-Changes*. Tech. rep. DOT HS 809 702. 3500 Transportation Research Plaza (0536), Blacksburg, VA 24061: DOT, Mar. 2004 (cit. on p. 10).

[96] H. Lenz, C. K. Wagner, and R. Sollacher. *Multi-anticipative car-following model*. In: *The European Physical Journal B – Condensed Matter and Complex Systems* 7 (1999), pp. 331–335. ISSN: 1434-6028. DOI: 10.1007/s100510050618 (cit. on p. 20).

[97] W. Levine and M. Athans. *On the optimal error regulation of a string of moving vehicles*. In: *Automatic Control, IEEE Transactions on* 11.3 (July 1966), pp. 355–361. ISSN: 0018-9286. DOI: 10.1109/TAC.1966.1098376 (cit. on p. 30).

[98] M. J. Lighthill and G. B. Whitham. *On kinematic waves. II. A theory of traffic flow on long crowded roads*. In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 229.1178 (1955), pp. 317–345. DOI: 10.1098/rspa.1955.0089 (cit. on p. 11).

[99] Mo Ye-Liu, He Hong-Di, Xue Yu, Shi Wei, and Lu Wei-Zhen. *Effect of multi-velocity-difference in traffic flow*. In: *Chinese Physics B* 17.12 (2008), p. 4446 (cit. on p. 20).

[100] Hong K. Lo. *A novel traffic signal control formulation*. In: *Transportation Research Part A: Policy and Practice* 33.6 (1999), pp. 433–448. ISSN: 0965-8564. DOI: 10.1016/S0965-8564(98)00049-4 (cit. on p. 205).

[101] Nicholas E. Lownes and Randy B. Machemehl. *VISSIM: a multi-parameter sensitivity analysis*. In: *Proceedings of the 38th Winter Simulation Conference (WSC)*. Monterey, California, 2006, pp. 1406–1413. ISBN: 1-4244-0501-7 (cit. on p. 71).

[102] Sigurdur F. Marinósson, Roland Chrobok, Andreas Pottmeier, Joachim Wahle, and Michael Schreckenberg. *Simulation des Autobahnverkehrs in NRW*. In: *SCS/ASIM – 16. Symposium in Rostock, Simulationstechnik*. Ed. by D. Tavangarian and R. Grützner. Rostock, Germany, 2002, pp. 517–523 (cit. on p. 29).

[103] Florian Mazur, Daniel Weber, Roland Chrobok, Sigurdur Freyr Hafstein, Andreas Pottmeier, and Michael Schreckenberg. *Basics of the Online Traffic Information System autobahn.NRW*. Hannovermesse 2005. Hannover, Germany, Apr. 2005 (cit. on p. 29).

[104] R.H. Middleton and J.H. Braslavsky. *String instability in classes of linear time invariant formation control with limited communication range*. In: *Automatic Control, IEEE Transactions on* 55.7 (July 2010), pp. 1519–1530. ISSN: 0018-9286. DOI: 10.1109/TAC.2010.2042318 (cit. on p. 31).

[105] E.A. Mueller. *Aspects of the history of traffic signals*. In: *Vehicular Technology, IEEE Transactions on* 19.1 (Feb. 1970), pp. 6–17. ISSN: 0018-9545. DOI: 10.1109/T-VT.1970.23426 (cit. on pp. 2, 25).

[106] Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer, eds. *Organic Computing – A Paradigm Shift for Complex Systems*. Springer Basel, 2011. ISBN: 978-3-0348-0130-0 (cit. on p. 3).

[107] Kai Nagel, Christopher Kayatz, and Peter Wagner. *Breakdown and recovery in traffic flow models*. In: *arXiv.org* (2001). arXiv:cond-mat/0112116 (cit. on p. 7).

[108] Kai Nagel and Michael Schreckenberg. *A cellular automaton model for freeway traffic*. In: *J. Phys. I France* 2 (Dec. 1992), pp. 2221–2229 (cit. on p. 13).

[109] L. Neubert, L. Santen, A. Schadschneider, and M. Schreckenberg. *Single-vehicle data of highway traffic: a statistical analysis*. In: *Phys. Rev. E* 60 (Dec. 1999), pp. 6480–6490. DOI: 10.1103/PhysRevE.60.6480 (cit. on pp. 5, 7, 9, 10, 151).

[110] G. F. Newell. *The rolling horizon scheme of traffic signal control*. In: *Transportation Research Part A: Policy and Practice* 32.1 (1998), pp. 39–44. ISSN: 0965-8564. DOI: 10.1016/S0965-8564(97)00017-7 (cit. on p. 205).

[111] Gábor Orosz, R. Eddie Wilson, and Bernd Krauskopf. *Global bifurcation investigation of an optimal velocity traffic model with driver reaction time.* In: *Phys. Rev. E* 70 (Aug. 2004), p. 026207. DOI: 10.1103/PhysRevE.70.026207 (cit. on p. 20).

[112] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Yibing Wang. *Review of road traffic control strategies.* In: *Proceedings of the IEEE* 91.12 (Dec. 2003), pp. 2043–2067. ISSN: 0018-9219. DOI: 10.1109/JPROC.2003.819610 (cit. on pp. 2, 25, 29, 205).

[113] Young-Jin Park and Frank F. Saccomanno. *Evaluating speed consistency between successive elements of a two-lane rural highway.* In: *Transportation Research Part A: Policy and Practice* 40.5 (2006), pp. 375–385. ISSN: 0965-8564. DOI: 10.1016/j.tra.2005.08.003 (cit. on p. 212).

[114] G. H. Peng and D. H. Sun. *A dynamical model of car-following with the consideration of the multiple information of preceding cars.* In: *Physics Letters A* 374.15–16 (2010), pp. 1694–1698. ISSN: 0375-9601. DOI: 10.1016/j.physleta.2010.02.020 (cit. on p. 20).

[115] Guang-Han Peng. *Stabilisation analysis of multiple car-following model in traffic flow.* In: *Chinese Physics B* 19.5 (2010), p. 056401 (cit. on p. 20).

[116] Guang-Han Peng and Di-Hua Sun. *Multiple car-following model of traffic flow and numerical simulation.* In: *Chinese Physics B* 18.12 (2009), p. 5420 (cit. on p. 20).

[117] L. Peppard. *String stability of relative-motion PID vehicle control systems.* In: *Automatic Control, IEEE Transactions on* 19.5 (Oct. 1974), pp. 579–581. ISSN: 0018-9286. DOI: 10.1109/TAC.1974.1100652 (cit. on p. 30).

[118] N. Rajewsky, L. Santen, A. Schadschneider, and M. Schreckenberg. *The asymmetric exclusion process: comparison of update procedures.* In: *Journal of Statistical Physics* 92 (1998), pp. 151–194. ISSN: 0022-4715. DOI: 10.1023/A:1023047703307 (cit. on p. 14).

[119] P. Richards. *Shock waves on the highway.* In: *Operations Research* 4.1 (Feb. 1956), pp. 42–51. DOI: 10.1287/opre.4.1.42 (cit. on p. 11).

[120] Urban Richter, Moez Mnif, Jürgen Branke, Christian Müller-Schloer, and Hartmut Schmeck. *Towards a generic observer/controller architecture for Organic Computing.* In: *Tagungsband der GI-Jahrestagung (Informatik).* Vol. 1. Dresden, Germany, Oct. 2006, pp. 112–119. ISBN: 973-3-88579-187-4 (cit. on p. 50).

[121] A. Schadschneider. *Traffic flow: a statistical physics point of view.* In: *Physica A: Statistical Mechanics and its Applications* 313.1–2 (2002), pp. 153–187. ISSN: 0378-4371. DOI: 10.1016/S0378-4371(02)01036-1 (cit. on p. 14).

[122] Andreas Schadschneider and Michael Schreckenberg. *Traffic flow models with 'slow-to-start' rules*. In: *Annalen der Physik* 509.7 (1997), pp. 541–551. ISSN: 1521-3889. DOI: 10.1002/andp.19975090703 (cit. on p. 14).

[123] Peter Schick. *Einfluss von Streckenbeeinflussungsanlagen auf die Kapazität von Autobahnabschnitten sowie die Stabilität des Verkehrsflusses*. PhD thesis. Universität Stuttgart, Institut für Straßen- und Verkehrswesen, June 2003. ISBN: 3-9808218-4-6 (cit. on pp. 27, 30).

[124] Hartmut Schmeck. *Organic Computing – a new vision for distributed embedded systems*. In: *Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, ISORC 2005*. Vol. 00. Los Alamitos, CA, USA: IEEE Computer Society, May 2005, pp. 201–203. ISBN: 0-7695-2356-0. DOI: 10.1109/ISORC.2005.42 (cit. on p. 3).

[125] Hilmar Schmundt. *Navigation – Geheimwaffe gegen den Stau*. In: *Spiegel* 49 (Dec. 2007), pp. 148–149 (cit. on p. 25).

[126] J. Schneider and A. Ebersbach. *Anticipatory drivers in the Nagel-Schreckenberg-model*. In: *International Journal of Modern Physics C* 13 (2002), pp. 107–113. DOI: 10.1142/S0129183102002985 (cit. on p. 14).

[127] Martin Schönhof and Dirk Helbing. *Empirical features of congested traffic states and their implications for traffic modeling*. In: *Transportation Science* 41.2 (2007), pp. 135–166. DOI: 10.1287/trsc.1070.0192 (cit. on pp. 7–9, 123).

[128] M. Schönhof, M. Treiber, A. Kesting, and D. Helbing. *Autonomous detection and anticipation of jam fronts from messages propagated by intervehicle communication*. In: *arXiv.org* (Nov. 2006). arXiv:physics/0611261 (cit. on p. 33).

[129] David Schrank, Bill Eisele, and Tim Lomax. *TTI's 2012 Urban Mobility Report*. Tech. rep. Texas A&M Transportation Institute, Dec. 2012 (cit. on p. 2).

[130] V. I. Shvetsov. *Mathematical modeling of traffic flows*. In: *Automation and Remote Control* 64 (11 2003), pp. 1651–1689. ISSN: 0005-1179. DOI: 10.1023/A:1027348026919 (cit. on p. 12).

[131] Martin Skutella. *An introduction to network flows over time*. In: *Research trends in combinatorial optimization*. Ed. by William Cook, László Lovász, and Jens Vygen. Springer Berlin Heidelberg, 2009, pp. 451–482. DOI: 10.1007/978-3-540-76796-1_21 (cit. on p. 40).

[132] Yuki Sugiyama, Minoru Fukui, Macoto Kikuchi, Katsuya Hasebe, Akihiro Nakayama, Katsuhiro Nishinari, Shin-ichi Tadaki, and Satoshi Yukawa. *Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam*. In: *New Journal of Physics* 10.3 (2008), p. 033001 (cit. on p. 74).

[133] Tie-Qiao Tang, Hai-Jun Huang, and Zi-You Gao. *Stability of the car-following model on two lanes*. In: *Phys. Rev. E* 72 (Dec. 2005), p. 066124. DOI: 10.1103/PhysRevE.72.066124 (cit. on p. 23).

[134] T. Q. Tang, H. J. Huang, S. G. Zhao, and G. Xu. *An extended OV model with consideration of driver's memory*. In: *International Journal of Modern Physics B* 23 (2009), pp. 743–752. DOI: 10.1142/S0217979209051966 (cit. on p. 20).

[135] TomTom. *TomTom European Congestion Index*. Tech. rep. TomTom International B.V., 2012 (cit. on p. 2).

[136] Peter Tondl, Klaus Jobmann, and Michael Meincke. *Dezentrale Verteilung sicherheitsrelevanter Verkehrsinformationen durch Fahrzeug-Fahrzeug-Kommunikation*. In: *11. Mobilfunktagung, Technologien und Anwendungen*. Osnabrück, 2006 (cit. on p. 33).

[137] Martin Treiber and Dirk Helbing. *Microsimulations of freeway traffic including control measures*. In: *Automatisierungstechnik* 49 (Nov. 2001), pp. 478–484. DOI: 10.1524/auto.2001.49.11.478 (cit. on pp. 7, 9, 27).

[138] Martin Treiber and Dirk Helbing. *Realistische Mikrosimulation von Straßenverkehr mit einem einfachen Modell*. In: *ASIM 2002, Tagungsband 16. Symposium Simulationstechnik*. Ed. by D. Tavangarian and R. Grützner. Rostock, 2002, pp. 514–520 (cit. on p. 23).

[139] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. *Congested traffic states in empirical observations and microscopic simulations*. In: *Phys. Rev. E* 62 (Aug. 2000), pp. 1805–1824. DOI: 10.1103/PhysRevE.62.1805 (cit. on pp. 18, 19).

[140] Martin Treiber, Arne Kesting, and Dirk Helbing. *Delays, inaccuracies and anticipation in microscopic traffic models*. In: *Physica A: Statistical Mechanics and its Applications* 360.1 (2006), pp. 71–88. ISSN: 0378-4371. DOI: 10.1016/j.physa.2005.05.001 (cit. on p. 19).

[141] Martin Treiber, Arne Kesting, and Dirk Helbing. *Understanding widely scattered traffic flows, the capacity drop, and platoons as effects of variance-driven time gaps*. In: *Phys. Rev. E* 74 (July 2006), p. 016123. DOI: 10.1103/PhysRevE.74.016123 (cit. on p. 3).

[142] Martin Treiber, Arne Kesting, and Christian Thiemann. *How much does traffic congestion increase fuel consumption and emissions? Applying fuel consumption model to NGSIM trajectory data*. In: *TRB 87th Annual Meeting Compendium*. Transportation Research Board, 2008 (cit. on p. 151).

[143] J. Treiterer and J. A. Myers. *The hysteresis phenomenon in traffic flow*. In: *Proceedings of the 6th International Symposium on Transportation and Traffic Theory*. Elsevier Publishing Company, Incorporated, Aug. 1974, pp. 13–38 (cit. on p. 5).

[144] Yohei Tutiya and Masahiro Kanai. *Exact shock solution of a coupled system of delay differential equations: a car-following model*. In: *J. Phys. Soc. Jpn.* 76 (Aug. 2007), 083002 (4 pages). DOI: 10.1143/JPSJ.76.083002 (cit. on p. 20).

[145] A. Vahidi and A. Eskandarian. *Research advances in intelligent collision avoidance and adaptive cruise control*. In: *Intelligent Transportation Systems, IEEE Transactions on* 4.3 (Sept. 2003), pp. 143–153. ISSN: 1524-9050. DOI: 10.1109/TITS.2003.821292 (cit. on p. 31).

[146] Tom Vanderbilt. *Traffic: Why We Drive the Way We Do (And What It Says about Us)*. Allen Lane, 2008. ISBN: 9780713999310 (cit. on p. 213).

[147] Pravin Varaiya. *Congestion, ramp metering and tolls*. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366.1872 (2008), pp. 1921–1930. DOI: 10.1098/rsta.2008.0015 (cit. on p. 30).

[148] Peter Wagner and Kai Nagel. *Comparing traffic flow models with different number of "phases"*. In: *The European Physical Journal B – Condensed Matter and Complex Systems* 63 (2008), pp. 315–320. ISSN: 1434-6028. DOI: 10.1140/epjb/e2008-00078-0 (cit. on pp. 12, 21).

[149] Haizhong Wang, Jia Li, Qian-Yong Chen, and Daiheng Ni. *Logistic modeling of the equilibrium speed-density relationship*. In: *Transportation Research Part A: Policy and Practice* 45.6 (2011), pp. 554–566. ISSN: 0965-8564. DOI: 10.1016/j.tra.2011.03.010 (cit. on p. 7).

[150] Ziyuan Wang, Lars Kulik, and Kotagiri Ramamohanarao. *Proactive traffic merging strategies for sensor-enabled cars*. In: *VANET '07: Proceedings of the Fourth ACM International Workshop on Vehicular Ad Hoc Networks*. New York, NY, USA: ACM, 2007, pp. 39–48. ISBN: 978-1-59593-739-1. DOI: 10.1145/1287748.1287755 (cit. on p. 34).

[151] Axel Wegener. *Organic-Computing-Konzepte und deren Umsetzung für dezentrale Anwendungen im Straßenverkehr*. PhD thesis. Universität zu Lübeck, 2009 (cit. on p. 38).

[152] Axel Wegener, Horst Hellbrück, Stefan Fischer, Björn Hendriks, Christiane Schmidt, and Sándor P. Fekete. *Designing a decentralized traffic information system – AutoNomos.* In: *Proceedings of the 16th ITG/GI – Fachtagung Kommunikation in Verteilten Systemen (KiVS).* Kassel, Germany, Mar. 2009 (cit. on p. 35).

[153] Axel Wegener, Horst Hellbrück, Stefan Fischer, Christiane Schmidt, and Sándor P. Fekete. *AutoCast: an adaptive data dissemination protocol for traffic information systems.* In: *Proceedings of the 66th IEEE Vehicular Technology Conference (VTC2007-Fall).* Baltimore, USA, Oct. 2007, pp. 1947–1951. DOI: 10.1109/VETECF.2007.409 (cit. on p. 38).

[154] Axel Wegener, Horst Hellbrück, Christian Wewetzer, and Andreas Lübke. *VANET simulation environment with feedback loop and its application to traffic light assistance.* In: *Proceedings of the 3rd IEEE Workshop on Automotive Networking and Applications.* New Orleans, LA, USA, Dec. 2008 (cit. on p. 53).

[155] Axel Wegener, Michał Piórkowski, Maxim Raya, Horst Hellbrück, Stefan Fischer, and Jean-Pierre Hubaux. *TraCI: an interface for coupling road traffic and network simulators.* In: *Proceedings of the 11th Communications and Networking Simulation Symposium (CNS).* Ottawa, Canada, Apr. 2008, pp. 155–163. ISBN: 1-56555-318-7 (cit. on p. 48).

[156] Axel Wegener, Elad M. Schiller, Horst Hellbrück, Sándor P. Fekete, and Stefan Fischer. *Hovering Data Clouds: a decentralized and self-organizing information system.* In: *Proceedings of the 1st International Workshop on Self-Organizing Systems.* Passau, Germany, Sept. 2006, pp. 243–247. DOI: 10.1007/11822035_22 (cit. on p. 35).

[157] Shi Wei, Chen Ning-Guo, and Xue Yu. *An asymptotic solvable multiple "look-ahead" model with multi-weight.* In: *Communications in Theoretical Physics* 48.6 (2007), p. 1088 (cit. on p. 20).

[158] Chen Wenjie, Chen Lifeng, Chen Zhanglong, and Tu Shiliang. *A realtime dynamic traffic control system based on wireless sensor network.* In: *International Conference Workshops on Parallel Processing (ICPP 2005).* June 2005, pp. 258–264. DOI: 10.1109/ICPPW.2005.16 (cit. on p. 205).

[159] Zhu Wen-Xing and Jia Lei. *Nonlinear analysis of a synthesized optimal velocity model for traffic flow.* In: *Communications in Theoretical Physics* 50.2 (2008), p. 505 (cit. on p. 20).

[160] Rainer Wiedemann. *Simulation des Straßenverkehrsflusses.* PhD thesis. Karlsruhe, 1974 (cit. on p. 13).

[161] R. E. Wilson, P. Berg, S. Hooper, and G. Lunt. *Many-neighbour interaction and non-locality in traffic models*. In: *The European Physical Journal B – Condensed Matter and Complex Systems* 39 (2004), pp. 397–408. ISSN: 1434-6028. DOI: 10.1140/epjb/e2004-00205-y (cit. on p. 20).

[162] R. Eddie Wilson. *An analysis of Gipps's car-following model of highway traffic*. In: *IMA Journal of Applied Mathematics* 66.5 (2001), pp. 509–537. DOI: 10.1093/imamat/66.5.509 (cit. on p. 190).

[163] R. Eddie Wilson. *Mechanisms for spatio-temporal pattern formation in highway traffic models*. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366.1872 (2008), pp. 2017–2032. DOI: 10.1098/rsta.2008.0018 (cit. on p. 190).

[164] Lars Wischhof, André Ebner, and Hermann Rohling. *Self-organizing traffic information system based on car-to-car communication: prototype implementation*. In: *Proceedings of the 1st International Workshop on Intelligent Transportation (WIT 2004)*. Hamburg, Germany, Mar. 2004 (cit. on p. 32).

[165] Lars Wischhof, André Ebner, and Hermann Rohling. *Information dissemination in self-organizing intervehicle networks*. In: *IEEE Transactions on Intelligent Transportation Systems* 6.1 (Mar. 2005), pp. 30–101. DOI: 10.1109/TITS.2004.842407 (cit. on p. 32).

[166] Lars Wischhof, André Ebner, Hermann Rohling, Matthias Lott, and Rüdiger Halfmann. *Adaptive broadcast for travel and traffic information distribution based on inter-vehicle communication*. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*. Columbus, Ohio, June 2003, pp. 6–11. DOI: 10.1109/IVS.2003.1212873 (cit. on p. 32).

[167] Lars Wischhof, André Ebner, Hermann Rohling, Matthias Lott, and Rüdiger Halfmann. *SOTIS – a self-organizing traffic information system*. In: *Proceedings of the 57th IEEE Vehicular Technology Conference (VTC 03 Spring)*. Jeju, South Korea, Apr. 2003 (cit. on p. 32).

[168] Tom de Wolf and Tom Holvoet. *Emergence versus self-organisation: different concepts but promising when combined*. In: *Engineering Self-Organising Systems*. Ed. by Sven A. Brueckner, Giovanna di Marzo Serugendo, Anthony Karageorgos, and Radhika Nagpal. Vol. 3464. Springer Berlin Heidelberg, 2005, pp. 1–15. ISBN: 978-3-540-26180-3. DOI: 10.1007/11494676_1 (cit. on pp. 3, 40).

[169] Ning Wu. *A new approach for modeling of Fundamental Diagrams*. In: *Transportation Research Part A: Policy and Practice* 36.10 (2002), pp. 867–884. ISSN: 0965-8564. DOI: 10.1016/S0965-8564(01)00043-X (cit. on p. 7).

[170] Hai Yang and Michael G. H. Bell. *A capacity paradox in network design and how to avoid it.* In: *Transportation Research Part A: Policy and Practice* 32.7 (1998), pp. 539–545. ISSN: 0965-8564. DOI: 10.1016/S0965-8564(98)00017-2 (cit. on p. 30).

[171] Hyejin Youn, Hawoong Jeong, and Michael T. Gastner. *The price of anarchy in transportation networks: efficiency and optimality control.* In: *arXiv.org* (Aug. 2008). arXiv:0712.1598 (cit. on p. 2).

[172] Lei Yu, Zhongke Shi, and Bingchang Zhou. *Kink–antikink density wave of an extended car-following model in a cooperative driving system.* In: *Communications in Nonlinear Science and Numerical Simulation* 13.10 (2008), pp. 2167–2176. ISSN: 1007-5704. DOI: 10.1016/j.cnsns.2007.07.008 (cit. on p. 20).

[173] Lei Zhang and David Levinson. *Ramp metering and freeway bottleneck capacity.* In: *Transportation Research Part A: Policy and Practice* 44.4 (2010), pp. 218–235. ISSN: 0965-8564. DOI: 10.1016/j.tra.2010.01.004 (cit. on p. 27).